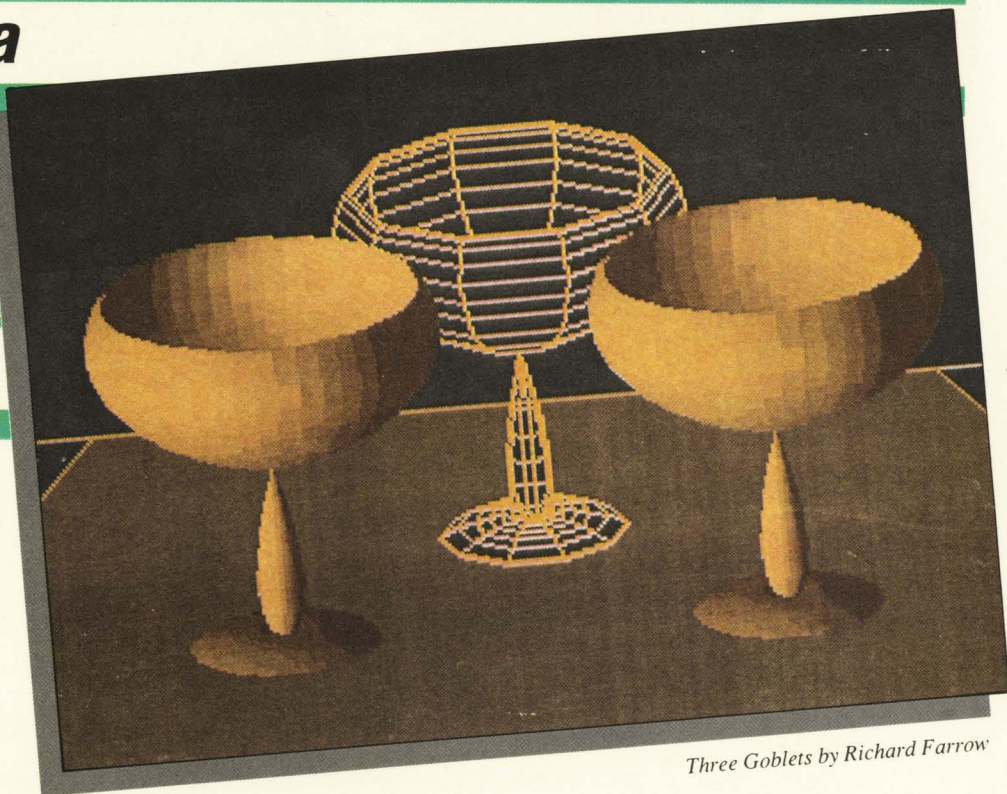
NUMERO 4 1989

LA RIVISTA PER I PROGRAMMATORI DI AMIGA

L. 7.000 - S.FR. 10,50

- Struttura di un programma Amiga
- Programma di supporto sviluppatori
- Transputer su Amiga • I dischi di Fred Fish
- I Device di Amiga • Prosound Designer
- Breakpoint, parte quarta • Amiga Structure Browser • I disk drive e le temporizzazioni
- Notizie Amiga
- Viewport
- Tips & Tricks

Three Goblets by Richard Farrow

TAXE PERCUE (TASSA RISCOSSA) MILANO FERR. CORR. SPEDIZIONE IN ABB. POST. GRUPPO IV/70

Dalla collaborazione con la rivista Compute!'s Amiga Resource, è nata l'Amiga che aspettavate! Chiara, aggiornatissima, più ricca, Amiga Magazine è la rivista ideale per il programmatore e adatta anche per i meno esperti. La nuovissima AMIGA MAGAZINE è già in edicola. Non perdetela !!

GRUPPO EDITORIALE JACKSON

Ladri!

Finalmente, è il caso di dirlo, mamma Commodore ha reso attivo anche in Italia il programma di supporto agli sviluppatori, cui avevamo accennato nel reportage dello scorso numero sulla conferenza tenutasi a Francoforte. L'evento in sé potrebbe anche passare sotto silenzio se non fosse che si tratta di un chiaro indicatore di una certa volontà positiva che, negli ultimi tempi, anima la Realcasa dell'home computer. Ebbene, il fatto che sia venuto il momento di codificare una sorta di regolamento dello sviluppo hardware e software intorno ad Amiga, deve essere considerato solamente un fatto positivo. Come ci si potrà rendere conto leggendo il resoconto relativo all'evento a pag. 15, il programma di supporto non è perfetto, ma permette anche a noi italiani di combattere ad armi, quasi, pari con il resto degli sviluppatori Amiga sparsi per il mondo.
Pur avendo infatti a disposizione gli stessi mezzi, ci resta da recuperare qualche anno di oscurantismo, qualche "briciolo" di cultura e soprattutto la mancanza di finanziamenti. Uno sviluppatore americano, inglese o tedesco, una volta che ha terminato il proprio prodotto, può cercare di piazzarlo sul proprio mercato, quello che conosce meglio e che gli è più facile raggiungere. Con i proventi derivanti dalle vendite in patria può cercare di espandere i propri orizzonti, distribuendo, magari anche adattando, il proprio prodotto in altri paesi. L'italiano, povero tapino, invece no. Non può sperare di avere alcun provento dagli utenti di Amiga del proprio paese. Pochi italiani hanno cercato fino a questo momento di distribuire propri prodotti in Italia e non si può certo dire che si siano arricchiti, né che abbiano recuperato i soldi spesi. Se a un produttore manca il supporto del proprio mercato cosa resta da fare, spostarsi su quello estero?
I costi per distribuire all'estero sono però nettamente superiori (basti pensare ai soli trasferimenti) e inoltre gli stranieri sono molto diffidenti su quanto, d'informatico, arrivi dall'Italia. Non è poi da sottovalutare il fatto che la nostra cultura è differente da quella di altri popoli: chi garantisce che un gioco sui Promessi Sposi (tanto per dire) possa aver successo in un paese al di là delle Alpi?
In sostanza, pur avendo a disposizione gli stessi mezzi, siamo distanti anni luce da una situazione minima che potrebbe garantire una vita florida a chi produce. Per uscire dalla situazione disastrosa in cui si trova il nostro paese non basta il '92, quando si realizzerà l'unione economica della CEE, ci vuole un cambiamento radicale nella mentalità degli utenti così come una legge seria e definitiva contro la pirateria software. I LADRI di software, come sarebbe più giusto chiamarli, togliendo quell'alone di romanticismo che gravita attorno alla figura del pirata, oltre ad aver rubato fior di miliardi ai legittimi proprietari, hanno rubato a tutti gli utenti e programmatori la possibilità di divertirsi e lavorare come tutti gli altri nel mondo.
Non si può infatti non addossare a questo malcostume (visto che non si può definire reato) la maggior parte delle responsabilità. Il peso di questa sconfitta nazionale, già visibile adesso, si pagherà più avanti quando sarà troppo tardi per porvi rimedio. Già ora fa rabbia vedere come paesi partiti, da un punto di vista informatico, forse più tardi di noi (tanto per non far nomi, la Francia) possano ora contare su strutture di sviluppo software di dimensioni ragguardevoli che esportano i loro prodotti, (tanto per non far nomi, la Infogrames francese), solo perché le loro produzioni sono tutelate.
Ben venga quindi il programma di supporto per gli sviluppatori, purché non sia la sola azione di un entusiasta in un mare di indifferenza e mafia.

Marco Ottolini

DIRETTORE RESPONSABILE
Paolo Reina

DIRETTORE EDITORIALE
Daniele Comboni

DIRETTORE TECNICO
Marco Ottolini

TRADUZIONI E REDAZIONE
Leonardo Fei, Stefano Simonelli,
Paolo Toccaceli, Giovanna Traversa

GRAFICA
Roberto Pessina

IMPAGINAZIONE ELETTRONICA
Roberto Gibertini

STAMPA
Grafica F.B.M. Gorgonzola (MI)

AUTORIZZAZIONE ALLA PUBBLICAZIONE
Trib. di Milano n. 866 del 20/12/88

AREA CONSUMER
PUBLISHER
Filippo Canavese

DIREZIONE
via Pola, 9 - 20124 MILANO
Tel. 02/69481
Telefax 02/6948238

REDAZIONE, PUBBLICITA,' AMMINISTRAZIONE, ABBONAMENTI
via Rossellini, 12 - 20124 Milano
Tel. 02/69481
Telex: 333436 GEJIT
Telefax: 02/6948489

CONCESSIONARIO ESCLUSIVO
SODIP - via Zuretti, 25 - 20125 Milano

MAGAZZINO
via Gasparotto, 15 - 20092 Cinisello B. (MI)
Tel. 02/61222527-6187736

SEDE LEGALE
via Pietro Mascagni, 14 - 20122 Milano

PREZZO DELLA RIVISTA: L. 7000
NUMERI ARRETRATI: L. 14000
ABBONAMENTO ANNUO (6 numeri): L. 34000
ESTERO: L. 68000

Spedizione in Abb. Gruppo IV/70

PUBLISHER
David H. Beatty

EDITOR
Mike Todd

CONTRIBUTING WRITERS
S. Ahlstrom, S. Ballantine, C.B. Blish, J. Butterfield, Betty Clay, D. Curtis, M. Dillon, A. Finkel, C. Innes, C. Gray, P. Kivolowitz, R. Mariani, B. Nesbitt, R. Peck, L. Phillips, B. Rakosky, J. Toebes, V.A. Wagner, D. Wood

Sommario

Associato al

Testata aderente al C.S.S.T. non soggetta a certificazione obbligatoria in quanto la presenza pubblicitaria è inferiore al 10%

Lettere...

Salve, vi scrivo per fare i complimenti alla vostra rivista che si occupa, finalmente, di una fetta di utenza sinora dimenticata, i programmatori. Ho letto l'articolo sul secondo numero della rivista che parlava di ARexx, un programma che mi interessa molto e sarei veramente intenzionato a reperire, ma una richiesta diretta negli USA, mi sembra proibitiva. È possibile acquistare questo programma in Italia? Mi sembrerebbe opportuno inserire negli articoli che parlano di prodotti stranieri se questi siano reperibili in Italia ed in caso affermativo a chi rivolgersi. Augurandovi di continuare su questa strada, vi saluto con affetto

Fabrizio P.

Il prodotto, se non indicato diversamente, non è da considerarsi direttamente disponibile in Italia. Questo perché la maggior parte dei programmi non lo sono. Purtroppo, è il caso di dirlo, nessuno pensa che programmi di questo tipo, cioè tutti quelli che non siano video giochi, possano avere un mercato anche in Italia. Ci sono due possibiltà: o ci si rassegna a provare a comprare all'estero, ringraziando i pirati, o si convincono i (pochissimi) distributori italiani che possono importare con fiducia anche in Italia. Naturalmente se qualcuno avesse già intrapreso questa strada ce lo faccia sapere al più presto in modo da informarne tutti i lettori.

...e Bit

Innanzitutto vorrei complimentarvi con voi, oltre che per la rivista, per l'apparizione, sul primo numero, di alcuni articoli che mi sono sembrati particolarmente interessanti. Il primo è quello di Leonardo Fei sui Virus: l'antivirus Guardian è veramente eccezionale!! Grazie a lui ho finito per sempre di combattere con i Virus e consiglio a tutti i possessori di Amiga 1000 di procurarsi il Guardian-creator per installare il Guardian direttamente sul dischetto del KickStart.
Un altro articolo che ha suscitato il mio interesse è 'Anteprima dell'AmigaDOS V1.3'. Leggendo infatti la parte dedicata al Fast File System e possedendo già il KickStart V1.3 ho deciso di fare alcune prove che considero interessanti e di cui ritengo opportuno informarvi: ho provato a installare il FFS sul drive esterno da 3"1/2 dell'Amiga. Ho configurato appositamente sul WB il file devs/mountlist chiamando DF3: il nuovo device, ho inserito un disco nel drive, ho impartito il comando 'MOUNT DF3:' dell'AmigaDOS V1.3, ho formattato il disco con l'opzione FFS (FORMAT DRIVE DF3: NAME prova FFS); ho infine aggiunto l'istruzione ADDBUFFERS DF3: 20 come da voi suggerito... ebbene?
Facendo attenzione a non estrarre il disco dal drive (il FFS funziona per ora solo con i device non rimuovibile come RAD e l'hard disk), tutto ha funzionato alla perfezione: ho effettuato numerose prove con il nuovo device e numerosi confronti con il drive DF0: con il file system normale ma ne ho tratto deludenti conclusioni: è vero che la velocità è notevolmente aumentata nella gestione della directory, dei comandi Dir, List, Delete, Rename, ma è invece rimasta identica nel caricamento dei programmi e nella gestione dei file sequenziali e relativi che sono le cose più importanti. È vero che è stata migliorata l'organizzazione dei dati sul dischetto, che è stato recuperato altro spazio, ma è anche vero che tutto ciò è andato a discapito della compatibilità: il dischetto non può essere più letto da nessun dispositivo sul quale non sia stato preventivamente montato il FFS nel modo in cui ho detto in precedenza, pena la comparsa del messaggio "Not a DOS disk in unit 1"; inoltre lo stesso dispositivo (DF3:) sul quale il FFS è stato installato non può più leggere un disco formattato in AmigaDOS senza l'opzione FFS, se non riferendosi (in questo caso) al device DF1:.
Spero che tutto ciò sia dovuto al fatto che il trackdisk.device sul KickStart non è predisposto (penso) al nuovo FFS, perché se questi inconvenienti non verranno risolti dalla nuova release del sistema operativo (V1.4) con la quale, si dice, il FFS sarà disponibile anche per i dischetti, si verranno a porre nuovi problemi di compatibilità per il software, per i copiatori, per il diskdoctor, per gli editor di tracce e per la maggior parte degli altri programmi di utilità generale che dovranno essere riadattati al nuovo sistema. Insomma, malgrado gli sforzi fatti dalla Commodore per ottimizzare le routine di gestione del drive, ritengo che il FFS non sia affatto superiore a un comune DISKARRANGER che, anche se effettua una velocizzazione solo su quanto già esistente sul disco, consente una lettura della directory e, in generale, una gestione del disco molto più efficace e veloce (basti guardare la rapida comparsa, da Workbench, delle icone dei file all'interno delle finestre), mantenendo la completa compatibilità con tutte le versioni dell'AmigaDOS!!!
Ecco ora cosa si deve aggiungere nel file DEVS:MountList:

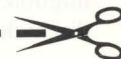
```
DF3:    Device = trackdisk.device
        FileSystem = L:FastFileSystem
        Unit = 1
        Flags = 1
        Surfaces = 2
        BlocksPerTrack = 11
        Reserved = 2
        Interleave = 0
        LowCyl = 0 ; HighCyl = 79
        Buffers = 20
        GlobVec = -1
        BufMemType = 3
        Mount = 1
        DosType = 0x444F5301
        StackSize = 4000
```

Se non si vuole usare il drive normalmente chiamato DF1: bisogna modificare il valore di Unit. Quindi un utente di Amiga 2000 che voglia usare il drive esterno dovrà scrivere Unit = 2, visto che normalmente quel drive si chiama DF2:.

Fabio Caruso, Messina

I Servizi
di

Amiga Transactor offre una serie di servizi per agevolare i propri lettori nel reperimento di software e materiale utile alla programmazione.
È disponibile l'intera libreria di dischetti di pubblico dominio curata da Fred Fish. Ogni dischetto contiene numerosi programmi e utility, spesso corredati da listati sorgenti e commenti degli autori.
Per districarsi fra le centinaia di programmi disponibili nei dischi di Fred Fish, è stato creato un apposito catalogo di 20 pagine. Tale elenco riporta, divisi per categoria e in ordine alfabetico, tutti i programmi presenti, completandoli con informazioni quali la descrizione della funzione, l'autore, il numero di versione, la disponibilità del sorgente e il disco nel quale sono contenuti. I dischetti possono essere ordinati contrassegnando i numeri desiderati, purché la quantità sia un multiplo di cinque. Per ordini amministrativi saremo costretti a non accettare ordini che non soddisfino questa regola. Tutto il materiale, a esclusione dei listati pubblicati sulla rivista, viene fornito nella versione originale americana, senza traduzione o modifiche.
A ogni numero della rivista corrisponde un disco chiamato "AmiTrans Disk" che contiene tutti i listati pubblicati su quel numero, nonché i corrispondenti eseguibili e tutti gli altri programmi di pubblico dominio menzionati negli articoli.

BUONO D'ORDINE

Completa il buono d'ordine (o una sua fotocopia) e spedire in busta chiusa a: I servizi di Transactor per Amiga - Via Rosellini, 12 - 20124 Milano

Si può allegare: assegno, contanti o fotocopia della ricevuta di versamento c/c n. 11666203 intestato a Gruppo Editoriale Jackson.
Non si effettuano spedizioni contrassegno

Desidero ricevere i seguenti articoli; contrassegnare con una X i numeri di Fish disk desiderati (a gruppi di 5)

Nota: Il disco 164 non è disponibile

☐ 1 ☐ 12 ☐ 23 ☐ 34 ☐ 45 ☐ 56 ☐ 67 ☐ 78 ☐ 89 ☐ 100 ☐ 111 ☐ 122 ☐ 133 ☐ 144 ☐ 155 ☐ 167
☐ 2 ☐ 13 ☐ 24 ☐ 35 ☐ 46 ☐ 57 ☐ 68 ☐ 79 ☐ 90 ☐ 101 ☐ 112 ☐ 123 ☐ 134 ☐ 145 ☐ 156 ☐ 168
☐ 3 ☐ 14 ☐ 25 ☐ 36 ☐ 47 ☐ 58 ☐ 69 ☐ 80 ☐ 91 ☐ 102 ☐ 113 ☐ 124 ☐ 135 ☐ 146 ☐ 157 ☐ 169
☐ 4 ☐ 15 ☐ 26 ☐ 37 ☐ 48 ☐ 59 ☐ 70 ☐ 81 ☐ 92 ☐ 103 ☐ 114 ☐ 125 ☐ 136 ☐ 147 ☐ 158 ☐ 170
☐ 5 ☐ 16 ☐ 27 ☐ 38 ☐ 49 ☐ 60 ☐ 71 ☐ 82 ☐ 93 ☐ 104 ☐ 115 ☐ 126 ☐ 137 ☐ 148 ☐ 159 ☐ 171
☐ 6 ☐ 17 ☐ 28 ☐ 39 ☐ 50 ☐ 61 ☐ 72 ☐ 83 ☐ 94 ☐ 105 ☐ 116 ☐ 127 ☐ 138 ☐ 149 ☐ 160 ☐ 172
☐ 7 ☐ 18 ☐ 29 ☐ 40 ☐ 51 ☐ 62 ☐ 73 ☐ 84 ☐ 95 ☐ 106 ☐ 117 ☐ 128 ☐ 139 ☐ 150 ☐ 161
☐ 8 ☐ 19 ☐ 30 ☐ 41 ☐ 52 ☐ 63 ☐ 74 ☐ 85 ☐ 96 ☐ 107 ☐ 118 ☐ 129 ☐ 140 ☐ 151 ☐ 162
☐ 9 ☐ 20 ☐ 31 ☐ 42 ☐ 53 ☐ 64 ☐ 75 ☐ 86 ☐ 97 ☐ 108 ☐ 119 ☐ 130 ☐ 141 ☐ 152 ☐ 163
☐ 10 ☐ 21 ☐ 32 ☐ 43 ☐ 54 ☐ 65 ☐ 76 ☐ 87 ☐ 98 ☐ 109 ☐ 120 ☐ 131 ☐ 142 ☐ 153 ☐ 165
☐ 11 ☐ 22 ☐ 33 ☐ 44 ☐ 55 ☐ 66 ☐ 77 ☐ 88 ☐ 99 ☐ 110 ☐ 121 ☐ 132 ☐ 143 ☐ 154 ☐ 166

DESCRIZIONE

☐ Fish Disk	Lit. 35.000 per gruppo di cinque
☐ Catalogo	Lit. 15.000 aggiornato fino al Fish 138
☐ AmiTrans Disk #1	Lit 15.000
☐ #2 #3 #4	Lit 15.000

Cognome ....

Nome ....

Via ....

Cap. .... Città ....

Prov. .... Tel. ....

Firma ....

(se minorenne quella di un genitore)
Gli ordini non firmati non verranno evasi.

Tutti i prezzi sono da intendersi IVA inclusa e spese di spedizione comprese.

I disk drive e le temporizzazioni

Gli errori più frequenti e le procedure corrette

di Bryce Nesbitt

Questo materiale arriva dalla Commodore-Amiga Inc. ed è apparso su un numero di AmigaMail (TM).

Parecchi tra gli sviluppatori software o hardware di aziende indipendenti impiegano i floppy disk drive in modo scorretto e realizzano cicli di attesa per le temporizzazioni in maniera del tutto inaccettabile.

Vorrei farvi presente che l'uso del trackdisk.device è sempre la procedura più sicura per accedere ai dischi; se, però, per qualche ragione avete proprio bisogno di operare direttamente sullo hardware o progettate hardware per i disk drive o avete bisogno di piccoli cicli di attesa, allora è opportuno che leggiate questo articolo. Troverete, infatti,una discussione sulla maniera migliore di creare ritardi e dieci punti che chi programma con i dischi deve tener ben presente.

La fonte di informazione più attendibile sulla temporizzazione dei drive è costituita dalla specifiche del costruttore. Questo articolo intende solamente porre in risalto gli errori più frequenti e grossolani. Rimane comunque consigliabile richiedere una copia delle specifiche tecniche di un disk drive e senz'altro la maggior parte dei distributori di componentistica non avrà problemi a farvela avere.

I dischi e il software

Questa parte è indirizzata principalmente alle persone che scrivono caricatori custom per i giochi. A causa di caricatori difettosi, ci sono migliaia di utenti Amiga che non possono caricare certi giochi e che pertanto NON POTRANNO MAI COMPRARLI.

Quindi...

1: Non partite da presupposti errati. Ad esempio, se è necessario che il motore del drive stia girando, accendetelo prima dell'uso. Non assumete che il codice di boot venga eseguito con il disk drive o il sistema in uno stato prevedibile a priori.

2: Non usate MAI un semplice ciclo con l'istruzione DBRA per la temporizzazione. C'è un gran numero di circostanze sotto le quali questo sistema non fornisce affatto risultati accurati. L'argomento verrà comunque ripreso in seguito, nel paragrafo *Come sprecare tempo.*

3: La linea STEP deve essere generalmente nello stato alto e deve essere attivata con un impulso basso. Prima, però, va predisposta la direzione con un'altra istruzione di scrittura nel registro opportuno. Un modo corretto è il seguente:

```
or.b    #%00000010,$bfd100  ;Predisponiamo
                              la direzione
and.b   #%11111110,$bfd100  ;Diamo un
                              impulso basso
nop                         ;Aspettiamo un
                              attimo
nop                         ; ...un altro
or.b    #%00000001,$bfd100  ;Riportiamo
                              alto STEP
;-- ora aspettiamo 3 millisecondi affinché
;-- la testina si porti sulla traccia
;-- successiva
```

La Commodore-Amiga ha sempre specificato che lo spostamento della testina alla traccia successiva avviene entro tre millisecondi. Ciò non toglie che alcuni drive possano farlo molto più rapidamente e che altri invece diano dei problemi già sotto i 2.8 millisecondi. Quando poi si cambia la direzione, bisogna aspettare un minimo di 18 millisecondi in totale per tener conto anche del tempo necessario per la stabilizzazione (settling time) della testina.

Si osservi inoltre che la linea TRACK ZERO non è valida fino a quando la testina non raggiunge effettivamente la traccia zero.

4: Quando fate partire il motore, aspettate che il segnale READY vada basso prima di leggere o scrivere; gli spostamenti invece sono del tutto leciti durante tale intervallo. Si noti che READY ha significato solo quando il segnale del motore è ON.

5: Per stabilire se c'è un disco nel drive, controllate il segnale DISKCHANGE; se tale segnale è basso, il disco è stato rimosso dopo l'ultimo controllo (e magari anche reinserito). In tal

caso muovete la testina per aggiornare il segnale e esaminate lo stato ottenuto dopo questa operazione.

6: Alcuni caricatori si servono di una traccia in più (o anche due) per mettervi dei dati o per le protezioni. Come Commodore-Amiga, non garantiamo che i drive che impiegheremo potranno sempre leggere più delle normali ottanta tracce. Diremo piuttosto che usare una traccia in più è abbastanza sicuro, due può ancora andare e tre è assolutamente un'idea molto azzardata.

7: Al termine di un'operazione DMA di scrittura sul disco, è necessario attendere 1.2 millisecondi prima di fare una qualsiasi operazione che coinvolga i drive (selezione, spostamento della testina e così via).

Nel tipo di disk drive montato sulle nostre macchine la testina che cancella e quella che scrive o legge sono di poco separate. Il drive mantiene attiva la testina di cancellazione fino a qualche istante dopo il termine della scrittura per compensare la presenza di questo "buco". Se non si attende il tempo sopra specificato si corre il rischio di scrivere su altre tracce o sull'altro lato del dischetto.

Per quanto riguarda la parte hardware...

8: La spia del drive non dovrebbe accendersi ad intermittenza durante l'accesso al disco. La spia dei nostri drive, infatti, specifica lo stato del motorino. Tipicamente lo stesso segnale è anche collegato alla linea IN_USE (pin 4).

9: Per ragioni di compatibilità con i modelli futuri, i drive si devono rifiutare di spostare la testina oltre la traccia zero. Se la testina è sulla traccia zero e viene ricevuto una richiesta di spostamento verso l'esterno, la testina non deve muoversi; in ogni caso, lo stato del segnale DISKCHANGE deve essere aggiornato, come avviene quando effettivamente la testina si sposta.

10: Le specifiche che devono essere sempre rispettate per i nostri drive da 90mm (3.5") sono le seguenti: 3ms per uno spostamento track-to-track, 15ms di tempo di stabilizzazione (settling time), allineamento radiale (misurato con il disco di allineamento Dysan) superiore all'80%, 500ms per l'avviamento del motore e 800ms per l'accensione completa del sistema.

Come sprecare tempo

La maniera di gran lunga peggiore per inserire un ritardo in un programma per Amiga è la seguente:

```
        move.w   #2000,d0
loop:   dbra     loop,d0
```

Eppure un gran numero di caricatori sfruttano sistemi simili per la temporizzazione dello spostamento della testina. Questo metodo è del tutto inaccettabile.

I programmi che usano questo metodo non funzionano già oggi su alcuni Amiga e probabilmente non funzioneranno su tutti i futuri modelli Amiga. Purtroppo sono diversi i casi di programmi che si rifiutano di caricare su macchine con particolari configurazioni.

Se si intende preservare il sistema operativo multitasking, il timer.device può essere impiegato per ottenere temporizzazioni che, nel frattempo, permettano ad altri task di girare. Se invece intendete impossessarvi della macchina, il metodo seguente è migliore:

```
loop:   btst.b   #0,$bfed01
        beq.s    loop
```

In questa maniera si impiega uno dei timer ad alta velocità. Come potrete rendervi conto dall'esempio riportato al termine dell'articolo, i timer sono molto semplici da usare.

Questo tipo di ciclo è migliore per vari aspetti; tra tutti spicca il semplice fatto che è più preciso.

Il primo metodo mostrato non produce temporizzazioni accurate in un gran numero di circostanze; la velocità, infatti, dipende dal tipo di CPU installata nel sistema, dal modo video, dal tipo di memoria in cui il programma risiede, dalle operazioni che il blitter sta magari compiendo, da quali interrupt sono attivi e da una varietà di altri fattori difficilmente controllabili.

Il metodo che impiega i timer è più accurato e può essere attivato e dimenticato. Il timer fa tutti i suoi bravi conteggi e intanto il vostro programma può proseguire liberamente. Se necessario, il timer può produrre un interrupt quando ha finito o può attivare un bit che potete andare a leggere quando volete. Si può inoltre fare in modo che il timer ricominci automaticamente a contare, una volta arrivato alla fine, così da produrre impulsi equidistanti, anche se il vostro software non riesce a rispondere immediatamente.

Come calcolare gli intervalli di tempo.

In primo luogo, vediamo alcune definizioni:

```
1 millisecondo (ms) = 1/1000 di secondo
1 microsecondo (us) = 1/1000000 di secondo
1 nanosecondo  (ns) = 1/1000000000 di secondo
```

Su un Amiga con un normale 68000 senza memoria aggiuntiva, l'istruzione DBRA di cui abbiamo parlato in precedenza esegue in circa 1.5 microsecondi. Il ciclo del nostro esempio equivale ad attendere 3000 (2000*1.5) microsecondi.

Ogni 8520 contiene al suo interno due timer a 16 bit che operano un conteggio alla rovescia alla frequenza di 0.709379 MHz (su una macchina PAL) cioè con un impulso di clock ogni 1.40968 microsecondi (con una macchina NTSC i valori sono rispettivamente di 0.715909 Mhz e 1.3968255 microsecondi); allora, per ottenere un ritardo di 3 millisecondi, dobbiamo inizializzare il contatore con il numero che si ottiene dividendo l'intervallo di tempo desiderato in microsecondi per 1.40968, cioè con 3000/1.40968 ovvero 2128.

Ecco un esempio completo dell'uso del timer:

```
; Un esempio completo di temporizzazione con
; l'8520. Fa accendere e spegnere
; la spia dell'alimentazione ogni 3 millisecondi.
; Questo programma prende
; brutalmente possesso della macchina, per cui fate
; attenzione!
;
; Le frequenze a cui oscillano i quarzi in Amiga
; sono: PAL  28.37516 MHz
;       NTSC 28.318181 MHz
;
; I due timer da 16 bit presenti in ciascuno degli
; 8520 contano ad un decimo
; della frequenza di clock della CPU (.709379 MHz
; in PAL). Ciò corrisponde ad
; un impulso ogni 1.40968 microsecondi. In una
; macchina NTSC la frequenza è
; leggermente superiore (.715909 MHz).
;
; Per aspettare 1/100 di secondo ovvero 10000
; microsecondi, il contatore-
; timer va inizializzato con il valore
; 10000/1.40968 = 7093.
;
; Per 3 millisecondi, invece, il valore richiesto è
; 3000/1.40968 = 2128
;
; Per ulteriori chiarimenti sugli 8520 consultate
; il Manuale hardware
;
ciaatalo    EQU $bfe401   ;Parte bassa del
                          ;Timer A
ciaatahi    EQU $bfe501   ;Parte alta del
                          ;Timer A
ciaaicr     EQU $bfed01   ;Interrupt
                          ;control  register
ciaacra     EQU $bfee01   ;Byte di controllo
                          ;del Timer A

            move.w    #$7fff,$dff09a  ;Blocca tutti
                  ;gli interrupt gestiti dai chip custom

; Inizializzazione; viene fatta solo una volta.
; Predisponiamo il Timer A in
; modo one-shot

            move.b    ciaacra,d0      ;Leggiamo il
                      ;registro di controllo di A
            and.b     #%11000000,d0   ;Non tocchiamo
                                 ;il flag 60/50Hz
            or.b      #%00001000,d0   ;o i bit di
                          ;direzione della seriale
            move.b    d0,ciaacra
            move.b    #%01111111,ciaaicr ;Spegniamo
                          ;tutti gli interrupt dell'8520

; Predisponiamo l'intervallo di tempo desiderato

            move.b    #(2128&255),ciaatalo ;mettiamo
                          ;gli 8 bit meno significativi
            move.b    #(2128>>8),ciaatahi ;e poi i
                          ;rimanenti nel contatore.

; Aspettiamo che il timer finisca

busy_wait:
            btst.b    #0,ciaaicr      ;Controlliamo
                          ;il flag di fine conteggio
            beq.s     busy_wait
            bchg.b    #1,$bfe001      ;Accendiamo o
                                 ;spegniamo la spia
            bset.b    #0,ciaacra      ;Facciamo
                                 ;ripartire il timer
            bra.s     busy_wait
```

Errata corrige

Per un banale errore, a pagina 68 del numero scorso, non era presente la figura ma solo la sua didascalia. La pubblichiamo ora, scusandoci con i lettori per l'accaduto.

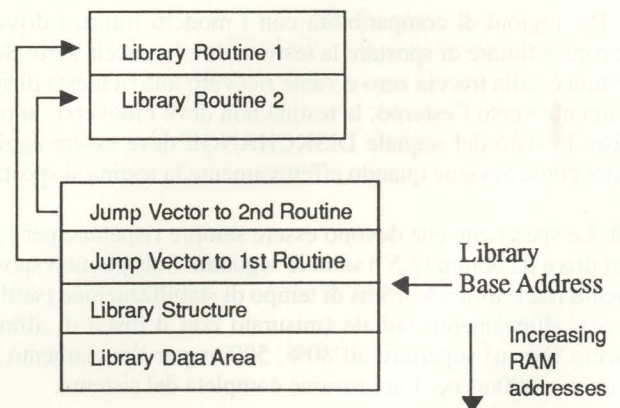



Figura 1: La relazione tra le componenti di una run-time library e le routine della library stessa.

La struttura di un programma Amiga

Diamo un'occhiata ravvicinata agli hunk

di Jim Butterfield

Jim Butterfield è uno scrittore, programmatore e professore di Toronto. Il suo interesse nei computer risale agli anni del KIM-1 da 1K, nei giorni primordiali della Commodore. L'abilità particolare di Jim nel trasmettere la sua conoscenza enciclopedica dei prodotti CBM attraverso articoli, libri, lezioni e programmi televisivi ha reso il suo nome famoso fra tutti gli utenti Commodore nel mondo.

Sembra che un programma per computer (spesso chiamato *tool* nel gergo del Workbench) sia una cosa semplice. Un programma viene spesso descritto come una serie di istruzioni che svolgono un determinato lavoro. In realtà ce n'è più di quanto non sembri dalla definizione.

Il nostro programma, chiamato *hunker*, ci permette di esaminare la complessa struttura dei programmi. È scritto in BASIC in modo che sia comprensibile a tutti e non sia necessario un costoso compilatore per farlo girare. Se volete vedere come funziona è molto semplice chiedere un LIST e vedere cosa succede.

I tre elementi

Un programma tipico è costituito da tre elementi. Primo, il codice: le istruzioni che fanno funzionare il programma. Successivamente, ci sono le costanti: valori fissi e messaggi usati dal programma. Infine, ci sono le variabili: valori che sono calcolati mentre il programma è in esecuzione.

Sembra che in BASIC questi tre tipi siano mischiati insieme. Un comando come PRINT J+5 contiene senza dubbio un'istruzione, ma contiene anche una costante (il valore 5). Sembra anche che contenga una variabile, J, ma in effetti ciò non è vero; il valore reale di J è memorizzato da qualche altra parte, separatamente dal programma.

Molto di quello che viene considerato *stile* nei programmi in linguaggio macchina è il risultato di una attenta pianificazione delle zone riservate alle costanti e di quelle riservate alle variabili.

Un programma stampa, per esempio, il messaggio ATTENDERE PER FAVORE in due momenti separati dell'esecuzione? Il programmatore immagazzinerà la stringa una volta sola e poi farà in modo che le due parti del programma vi accedano. Bisogna mettere a zero il contenuto di quindici variabili a un certo punto del programma? Il programmatore probabilmente le metterà una vicina all'altra e affiderà il compito di metterle a zero contemporaneamente a un semplice loop che pulisce la memoria. Spesso è vero che si investe molto più tempo nel pianificare le strutture dati che nello scrivere il codice vero e proprio.

La struttura dei programmi per Amiga è stata concepita in origine per tenere questi tre elementi separati, suddividendoli in hunk (pezzi) separati. Ci sono quindi tre tipi di hunk: le istruzioni vengono memorizzate in un *code hunk*, le costanti in un *data hunk* e le zone riservate alle variabili in un *BSS hunk* (Base of Stack Segment = inizio della zona di stack). Quest'ultimo tipo di hunk, quando viene memorizzato su disco, non contiene dati, ma solo le dimensioni dello spazio che sarà necessario riservare al momento del caricamento in memoria. Le variabili che ricadono in quest'ultimo tipo di hunk verranno inizializzate dal programma mentre questo sarà in esecuzione.

BSS è una sigla che viene dal mondo UNIX e le parole Base of Stack Segment non hanno alcun significato pratico su Amiga. Preferirei usare una traduzione ben più vecchia, proveniente dalle macchine precedenti, e pensare a BSS come Block Segment Size (dimensioni del segmento).

In origine fu stabilito che i programmi avrebbero dovuto contenere uno per ciascuno dei tre tipi di hunk. In realtà ce ne sono di più; molti programmi sono composti a loro volta da programmi indipendenti o da routine collegate insieme. Vogliamo fare qualche calcolo? I programmi per svolgere questo compito verranno probabilmente richiamati separatamente (hunk e tutto il resto) e collegati (tramite il linker) al nostro codice. Il nostro programma, inoltre, potrebbe essere preceduto da del codice di startup già pronto, generalmente inserito in maniera automatica dal compilatore. È molto facile che un comune programma finisca per essere composto di almeno una dozzina o più hunk. Come potete immaginare, il programmatore principiante, che utilizzi il C o il linguaggio macchina, potrebbe avere qualche problema a trovare il suo piccolo pezzetto di codice in mezzo alle masse di materiale extra che sono state aggiunte.

Il caricamento in ordine sparso (scatter loading)

Sui computer più semplici (per esempio il C64), un programma è composto da un pezzo unico. L'operazione di caricamento trasferisce questo pezzo in una zona di memoria e il programmatore spesso sa anche in anticipo dove si trovi esattamente questa zona.

Su Amiga la storia è differente. Un programma può essere costituito da un gran numero di hunk. Il programma verrà caricato in *ordine sparso*, dove questo significa che ogni hunk verrà sistemato individualmente, durante il caricamento, in una parte della memoria disponibile completamente separata dalle altre. Il programma di caricamento che sovrintende a questa operazione si preoccuperà anche di stabilire i collegamenti necessari fra i vari hunk.

Noi non dobbiamo preoccuparci di nessuna di queste cose. Amiga si prende cura di tutti i dettagli al posto nostro e noi non ci renderemo neanche conto che il nostro programma è frammentato in dozzine di pezzi sparpagliati per tutta la memoria. Tutto funziona insieme magnificamente.

Ogni volta che un hunk fa riferimento a un altro hunk, per prendere o depositare dei dati o per chiamare una subroutine, questo fatto viene annotato in una *tabella di rilocazione* (relocation table). La tabella aiuta il programma caricatore (loader) a sistemare gli indirizzi in modo che puntino al posto giusto. Questo è una dettaglio tecnico: dal momento che tutti gli indirizzi usati in questo modo sono contenuti in *longword* da 32 bit, la tabella viene chiamata RELOC_32. Vedremo parecchie di queste tabelle quando proveremo il programma *hunker*.

Perché usare gli hunk?

Questo tipo di caricamento in ordine sparso presenta molti vantaggi. Il fattore più importante è il migliore utilizzo della memoria. Supponiamo, per esempio, di avere un programma di 50K. Se fosse composto unicamente di un pezzo, dovremmo trovare 50K di memoria libera contigua per poterlo caricare. Ma se questo programma fosse composto da dieci hunk, ognuno di 5K, avremmo molti meno problemi a trovare lo spazio per caricarli tutti.

Ci sono due tipi di memoria in Amiga: *chip* e *fast* (a meno che non possediate un A2000 con 1 Mega o una espansione per A500, avrete normalmente a disposizione solo 512K di memoria chip). I dati che vengono utilizzati dai chip audio/video (Paula-Agnus-Denise o PAD) devono essere scritti in memoria chip, in quanto questa zona è l'unica che i chip possono 'vedere'. Tutto quello che non deve essere utilizzato da questi chip è meglio che venga scritto in memoria fast, se questa è disponibile, in quanto quest'ultima è più veloce.

Ogni hunk di un programma può essere etichettato a seconda del tipo di memoria nel quale dovrà essere caricato. Se un hunk viene etichettato CHIP, per esempio, verrà caricato solo in memoria chip. Se, invece, l'hunk non contiene indicazioni specifiche sulla sua destinazione, verrà caricato in memoria fast, ove fosse disponibile, oppure in memoria chip. Nel raro caso in cui un hunk sia marchiato FAST, questo verrà caricato esclusivamente in tale tipo di memoria. Un programma contenente questo tipo di hunk non può girare su un Amiga provvisto solo di 512K di chip ram.

Possiamo a questo punto immaginare cosa succede quando un grosso programma (diciamo 200K) viene caricato in un Amiga dotato di espansione di memoria. Alcuni hunk, quelli classificati CHIP, vengono caricati in memoria chip, mentre tutti gli altri vengono caricati nella memoria fast. Risultato: la memoria chip non viene sprecata, lasciandone una buona parte libera, e il programma gira più velocemente in quanto le istruzioni risiedono in memoria fast.

Si parte!

Digitate il programma hunker; è scritto in AmigaBASIC. Salvatelo e date il RUN. Possiamo, a questo punto, specificare il nome di qualsiasi programma eseguibile. Il comando LINE INPUT ci permette di usare i caratteri come il due punti o la virgola senza creare confusione nell'interprete BASIC. Uno dei miei programmi favoriti è:

```
DF0:c/BindDrivers
```

Possiamo dare molte risposte alla domanda del programma VUOI VEDERE I BYTE? Rispondendo YES vedremo la stampa dei byte contenuti, mentre NO produrrà solo la lista degli hunk presenti. Rispondendo D o DATA otterremo la stampa dei soli hunk di tipo DATA (dove potremmo riconoscere le stringhe del programma) mentre A o ALL produrrà anche il contenuto delle relocation table. In generale queste ultime due opzioni non sono molto utili.

Potreste essere sorpresi nel vedere qualcos'altro in alcuni programmi: i *simboli*. Questi sono i nomi usati dal programmatore o dal sistema operativo per identificare determinate parti di codice. Questi simboli non sono più necessari dopo che le operazioni di compilazione/assemblamento e link sono state effettuate. Sono lì solo per aiutare nel debugging del programma.

Durante le vostre esplorazioni di vari programmi e comandi, vi meraviglierete di certo per le grandi differenze nel numero di hunk che compongono un programma. Alcuni piccoli programmi (come BindDrivers) usano molti hunk. Alcuni programmi di dimensioni medie (come Echo) non ne utilizzano quasi nessuno. Il programmatore controlla gli hunk prodotti durante la compilazione e il link del proprio programma. In fondo si tratta anche di una faccenda di stile.

Buona ricerca!

Abbiamo forse bisogno di sapere quello che è stato appena detto? Non direttamente. I programmi continueranno a funzionare bene sia che noi si sappia o meno dell'esistenza degli hunk.

In ogni caso, è sempre interessante vedere il funzionamento interno del nostro computer. Una comprensione migliore sulla struttura di un programma, prima che questo venga caricato

dalla macchina, può darci più idee su come usare un debugger (come WACK) per vederne il funzionamento interno.

Le idee che trarremo giocando con HUNKER ci aiuteranno a prepararci al giorno in cui ci tufferemo in linguaggi di programmazione più avanzati e in una più seria programmazione di Amiga.

```
PRINT "Hunk Analysis Program   -  J Butterfield"
PRINT "v0.2   December 88"
PRINT
PRINT "nome del file eseguibile? ";
LINE INPUT f$
 OPEN f$ FOR INPUT AS #1
 length=LOF(1)
 PRINT f$;":";length;"bytes:"
 IF length>8 THEN
  h$=INPUT$(4,1)
  GOSUB Eval
  IF Valu&=1011 THEN  'hunk header
    PRINT "vuoi vedere i byte? ";
    LINE INPUT x$
    'view = 0 non visualizza i byte
    'view = 1 byte di dati e hex
    'view = 2 tutti i byte, solo dati hex
    'view = 3 tutti i byte, tutti gli hex
    View=2
    IF x$="n" OR x$="no" THEN View=0
    IF x$="d" OR x$="data" THEN View=1
    IF x$="a" OR x$="all" THEN View=3
    PRINT "mando l'output alla stampante? ";
    LINE INPUT x$
    dev$="scrn:"
    IF x$="y" OR x$="yes" THEN dev$="prt:"
    OPEN dev$ FOR OUTPUT AS #2
    PRINT#2,"File:";f$
    h$=INPUT$(4,1)
    GOSUB Eval
    unl=Valu&
    WHILE unl>0
     PRINT#2,"Nome dell'unita' = ";INPUT$(unl*4,1)
     h$=INPUT$(4,1)
     GOSUB Eval
     unl=Valu&
    WEND
   PRINT#2,"HUNK HEADER  ";
   showcnt=0
    h$=INPUT$(4,1)
    GOSUB Eval
    size=Valu&
    PRINT#2,"Dimensioni";size;
    h$=INPUT$(4,1)
    GOSUB Eval
    Hunk=Valu&
    PRINT#2," Da";Hunk;
    h$=INPUT$(4,1)
    GOSUB Eval
    l=Valu&
 PRINT#2,"a";l
Ask=0:IF View>1 THEN Ask=1
GOSUB Dump

WHILE NOT EOF(1)
 t$=INPUT$(1,1)
 h$=INPUT$(3,1)
 HunkType$=""
 p=ASC(t$)/64
 IF p<>INT(p) THEN PRINT "?"
 IF p=1 THEN HunkType$=".CHIP"
 IF p=2 THEN HunkType$=".FAST"
 IF p=3 THEN HunkType$=".???"
 GOSUB Eval
 IF Valu&=1001 OR Valu&=1002 THEN
  PRINT#2,"Hunk";Hunk;
  Hunk=Hunk+1
  IF Valu&=1001 THEN
     Ask=1:PRINT#2,"HUNK.CODE";HunkType$
  END IF
  IF Valu&=1002 THEN
     Ask=2:PRINT#2,"HUNK.DATA";HunkType$
  END IF
  h$=INPUT$(4,1)
  GOSUB Eval
  size=Valu&
  IF View=0 OR View=3 THEN Ask=View
  GOSUB Dump
 ELSEIF Valu&=1003 THEN
  PRINT#2,"Hunk";Hunk;
  Hunk=Hunk+1
  h$=INPUT$(4,1)
  GOSUB Eval
  PRINT#2,"HUNK.BSS.";pub$;" [ ";size;
                                  "long words,";size*4;"bytes ]"
 ELSEIF Valu&=1004 THEN
  h$=INPUT$(4,1)
  GOSUB Eval
  WHILE Valu&<>0
   size=Valu&
   h$=INPUT$(4,1)
   GOSUB Eval
   hk=Valu&
   PRINT#2,"..HUNK.RELOC_32   Hunk";hk;
   IF View<2 THEN
    Ask=0
   ELSE
    PRINT#2,"vettori:"
    Ask=1
   END IF
   GOSUB Dump
   h$=INPUT$(4,1)
   GOSUB Eval
  WEND

 ELSEIF Valu&=1008 THEN
  PRINT#2,"..HUNK.SYMBOL"
  h$=INPUT$(4,1)
  GOSUB Eval
```

```
     WHILE Valu&>0
      length=Valu&*4
      nam$=INPUT$(length,1)
      WHILE MID$(nam$,length,1)=CHR$(0)
       length=length-1
      WEND
      PRINT#2," [";LEFT$(nam$,length);"]  ";
                                    SPACE$(30-length);
      h$=INPUT$(4,1)
      showcnt=0
      GOSUB nshow
      PRINT#2,
      h$=INPUT$(4,1)
      GOSUB Eval
    WEND

    ELSEIF Valu&=1010 THEN
     PRINT#2,"..HUNK.END"
    ELSEIF Valu&=1013 THEN
     ' not sure about this one
     PRINT#2,"..HUNK.OVERLAY"
     h$=INPUT$(4,1)
     GOSUB Eval
     Ask=View:IF Ask>1 THEN Ask=Ask-1
     size=Valu&
     GOSUB Dump
    ELSEIF Valu&=1014 THEN
     PRINT#2,"..HUNK.BREAK"
    ELSE
     CLOSE #2
     OPEN "scrn:" FOR OUTPUT AS #2
     PRINT Valu&
     PRINT "diagnostica:"
     FOR j=1 TO 12
      h$=INPUT$(4,1)
      GOSUB nshow
     NEXT j
     STOP
    END IF
   WEND
   CLOSE #2
  ELSE
   PRINT "non e' un programma eseguibile"
  END IF
 ELSE
  PRINT "troppo corto"
 END IF
 CLOSE #1
 PRINT
END

Dump:
   showcnt=0:tx$=" "
   FOR j=1 TO size
    h$=INPUT$(4,1)
    IF Ask>0 THEN
     IF showcnt>4 THEN
      showcnt=0
      PRINT#2,tx$
      tx$=" "
     END IF
     PRINT#2," ";
     FOR k=1 TO 4
      k0=ASC(MID$(h$,k,1))
      IF Ask>1 THEN
       IF k0>31 AND k0<127 THEN
        tx$=tx$+CHR$(k0)
       ELSE
        tx$=tx$+"."
       END IF
      END IF
      k%=INT(k0/16)
      GOSUB byteshow
      k%=k0-k%*16
      GOSUB byteshow
     NEXT k
     showcnt=showcnt+1
    END IF
   NEXT j
   IF Ask>0 THEN
    WHILE showcnt<=4
      PRINT#2,SPACE$(9);
      showcnt=showcnt+1
    WEND
    PRINT#2,tx$
   ELSE
    PRINT#2," [ ";size;"long words,";size*4;
                                          "bytes ]"
   END IF
RETURN

nshow:
   PRINT#2," ";
   PRINT#2,"[";
   FOR k=1 TO 4
    k0=ASC(MID$(h$,k,1))
    k%=INT(k0/16)
    GOSUB byteshow
    k%=k0-k%*16
    GOSUB byteshow
   NEXT k
   PRINT#2,"]";
RETURN

byteshow:
   t%=k%:IF t%>9 THEN t%=t%+7
   PRINT#2,CHR$(t%+48);
RETURN

Eval:
   Valu&=0
   FOR k=1 TO LEN(h$)
    Valu&=Valu&*256&+ASC(MID$(h$,k,1))
   NEXT k
RETURN
```

Programma di supporto agli sviluppatori

Commodore Italiana

Riceviamo direttamente dalla Commodore Italiana il testo che annuncia l'inizio del piano di supporto agli sviluppatori Amiga cui avevamo accennato nello scorso numero. Riportiamo il testo integralmente ponendo solo un breve commento alla fine.

Il concetto

Nell'ottica di promuovere lo sviluppo di software, schede aggiuntive e periferiche per Amiga, Commodore Italiana ha implementato un Programma Supporto Sviluppatori, sotto il diretto patrocinio di Commodore International.

Lo scopo di questo servizio è quello di mettere a disposizione delle società o dei singoli impegnati attivamente nello sviluppo HW/SW per Amiga un ambiente che fornisca tutti gli elementi di supporto necessari:

- Una pronta e completa diffusione di informazioni tecniche
- la possibilità di acquistare materiale a prezzi particolari
- dei mezzi di comunicazione rapidi ed efficaci
- l'accesso diretto a nuove versioni di software e hardware nel periodo di beta-testing
- la creazione e il coordinamento di contatti con editori italiani e stranieri

Questo Programma di Supporto viene implementato a livello internazionale da Commodore International, che ha stabilito una rete di collegamento tra tutte le filiali Commodore nel mondo, le quali offriranno il loro proprio servizio di supporto locale. Tutti gli ambiti locali sono quindi collegati tramite mezzi elettronici tra loro e con i relativi uffici guida europei e americani. Questa rete permette quindi un contatto privilegiato e diretto con le persone che hanno progettato l'Amiga e il suo System software, in modo da poter rapidamente rispondere ai quesiti tecnici.

I servizi offerti da Commodore

<u>Abbonamento ad Amiga Mail e Amiga Mail Market</u>

Queste newsletter si occupano rispettivamente degli aspetti tecnici e di diffusione prodotti, con cadenza bimestrale.
Già note nel mondo per la loro peculiarità, riportano informazioni spesso vitali per gli sviluppatori.

<u>Uno sconto sugli acquisti</u>

Per tutti i prodotti della linea Amiga verrà praticato uno sconto differenziato a seconda della categoria di appartenenza degli sviluppatori. Le uniche condizioni poste sono:

- impegno a non rivendere il materiale nei 12 mesi seguenti la data d'acquisto
- acquisto massimo di 2 pezzi per tipo per anno fiscale
- pagamento contrassegno

<u>Accesso alla libreria sviluppatori</u>

Commodore fornirà una libreria di dischetti con i migliori esempi ed utility di pubblico dominio, possibilmente con i sorgenti.

<u>Supporto diretto</u>

Possibilità di contatto diretto telefonico per quesiti che richiedano risposte urgenti. In particolare la possibilità di avere in breve tempo informazioni tecniche specifiche grazie al contatto privilegiato con l'Engineering.

<u>Documentazione sul Sistema Operativo</u>

Uno sconto sull'acquisto della documentazione di riferimento del Sistema Operativo:

```
- AW ROM Kernel: Exec
- AW ROM Kernel: Libraries & Devices
- AW ROM Kernel: Includes & Autodocs
                 revised & updated

- A500/A2000 Technical Reference Manual
```

Accesso all'ulteriore documentazione tecnica:

```
- Software Toolkit
- IFF Manual & disks
- Note delle Developer's Conferences
- AmigaDOS 1.3 Native Developer Kit
```

<u>Documentazione tecnica hardware</u>

Possibilità di acquisto di manuali per l'assistenza, schemi e bollettini tecnici.

Parti di ricambio

Possibilità di acquisto di parti di ricambio custom con lo sconto del 15% sul prezzo di listino, a queste condizioni:

- limitato alle parti relative al progetto in corso di sviluppo
- impegno a non rivendere il materiale
- quantitativi limitati e comunque soggetti a verifica da parte di Commodore Italiana
- pagamento contrassegno

Beta-Test Software

Commodore migliora incessantemente la funzionalità del Sistema Operativo. Nell'ottica di verificare la compatibilità dei nuovi progetti e di orientare lo sviluppo in modo da trarre il massimo profitto da nuove versioni del System Software, sarà consentito l'accesso alle versioni Alpha e Beta. Naturalmente, la disponibilità del software sopracitato sarà subordinata ad un formale impegno di non divulgazione e di rapida informazione verso Commodore dei risultati di questi test.

Beta-Test Hardware

Commodore mette a disposizione degli sviluppatori la possibilità di accedere a nuovi modelli di Amiga e di periferiche in modo da poter testare il nuovo hardware e indirizzare i nuovi progetti in accordo alle piattaforme disponibili in futuro. La prova di queste apparecchiature sarà fatta presso la sede della Commodore Italiana su appuntamento, e sarà soggetta ad impegno di non divulgazione.

Nei limiti del possibile, e quando la situazione lo richieda effettivamente, Commodore Italiana potrà prestare per un periodo limitato l'hardware in beta-testing.

Accesso gratuito a BIX

BYTE Information Exchange (BIX) è il mezzo di comunicazione privilegiato degli sviluppatori Amiga. Agli sviluppatori commerciali viene offerta gratuitamente l'opportunità di partecipare alla conference amiga.com, condotta da CATS (Commodore-Amiga Technical Support).

Supporto marketing

Per i progetti hardware e software più interessanti, Commodore potrà fare da tramite di aiuto e collegamento con editori e produttori italiani e stranieri, nei limiti imposti da opportunità e disponibilità.

Altri servizi

Gli sviluppatori aderenti al Programma di Supporto saranno ufficialmente invitati alle conferenze che periodicamente si terranno in Italia e all'estero, usufruendo di uno sconto particolare.

Ciò che Commodore si aspetta dagli sviluppatori

Una volta iscritti come sviluppatori, si è tenuti a rispettare certi obblighi in contropartita ai servizi offerti:

- Tenere al corrente Commodore Italiana dello stato di avanzamento dei progetti, con cadenza mensile.
- In caso si testino versioni preliminari di software e/o hardware, informare tempestivamente Commodore Italiana dei risultati di questi test, con rapporti scritti dettagliati.
- Una volta terminato un progetto, inviare tre campioni a Commodore Italiana, in modo da tenere un catalogo aggiornato dei prodotti disponibili e da poterli diffondere ad altre filiali Commodore nel mondo. Naturalmente Commodore Italiana e le altre filiali si impegnano a non diffondere i prodotti ad altre società o singoli senza la preventiva autorizzazione scritta dello sviluppatore.
- Impegno a non rivendere l'hardware acquistato per lo sviluppo nei 12 mesi seguenti l'acquisto, salvo diversi accordi scritti.
- Impegno a rispettare la confidenzialità delle informazioni e a non divulgare a terzi le suddette informazioni.
- Impegno a rispettare le linee direttrici emesse da Commodore per lo sviluppo del software applicativo e dell'hardware aggiuntivo.

Lo sviluppatore resta comunque unico proprietario e responsabile dei suoi progetti. Commodore non avrà alcun diritto su di essi.

L'organizzazione

È intenzione di Commodore incoraggiare tutti gli sviluppatori, dalla società di sviluppo in grado di commercializzare i suoi prodotti all'hobbyista con idee geniali.

Come si può vedere dal sommario dei servizi offerti da Commodore, alcuni di questi sono particolarmente onerosi e saranno quindi riservati a gruppi selezionati.

A questo scopo, sono state create tre categorie di sviluppatori: commerciali, certificati e registrati. Commodore deciderà a quale categoria un dato sviluppatore appartiene sulla base delle informazioni fornite, dei progetti correnti e futuri e della struttura commerciale.

Lo sviluppatore commerciale è una società che ha già dimostrato le capacità di concepire e commercializzare un prodotto, sì a direttamente che tramite distributori. Potranno essere accettati in questa categoria anche sviluppatori con progetti in applicazioni chiave, di importanza strategica per Commodore. I problemi e le richieste poste da questi sviluppatori saranno trattati con la massima priorità.

Lo sviluppatore certificato è una società o un singolo che, pur

non commercializzando direttamente un prodotto, lavora comunque su progetti interessanti con potenzialità commerciali. In questa categoria possono essere compresi università o club o gruppi di utenti avanzati (User Groups). Gli sviluppatori certificati non beneficeranno di supporto telefonico.

Lo sviluppatore registrato è una società o un singolo interessata/o a progetti su Amiga senza necessariamente avere scopi commerciali. Gli sviluppatori registrati non beneficeranno di supporto telefonico.

In seguito all'esame della documentazione relativa alla richiesta di partecipazione al Programma di Supporto Sviluppatori Amiga, Commodore Italiana valuterà l'opportunità di inserire la richiesta stessa nella lista degli sviluppatori "pre-recommended", che sarà poi eventualmente ratificata da Commodore International, ritornando il numero di iscrizione internazionale.

Il comunicato sembra sufficientemente chiaro e non dovrebbe dare origine a incomprensioni. Comunque ci preme sottolineare un punto che forse non è stato trattato a sufficienza.
Lo sviluppatore, hardware o software che sia, una volta che ha deciso di partecipare all'operazione deve accordarsi con Commodore per presentare il proprio progetto. In base alle informazioni ottenute, Commodore International può decidere se accettare la domanda ed eventualmente per quale delle tre categorie. Nessuno si metta quindi in mente che pagando $500 possa avere il diritto di appartenere alla categoria degli sviluppatori commerciali, deciderà solo Commodore.
I prezzi praticati ci sembrano corretti, eccezion fatta per quello dello sviluppatore registrato. In particolare è la figura stesso dello sviluppatore registrato che ci lascia perplessi: non ha diritto quasi a nulla e quel poco lo deve anche pagare extra.
La barriera formata dai prezzi da corrispondere a Commodore International dovrebbe essere già sufficiente a tenere lontano chi non è realmente interessato, ma, se tutto ciò non bastasse, ci sarebbe comunque lo scoglio dell'esame da parte di Commodore per tenere fuori dalla porta i perditempo e gli scocciatori.
Inoltre per garantire un lavoro tranquillo agli sviluppatori seri occorrerebbe fare piazza pulita una volta per tutte di tutte quelle persone che gravitano "nell'ambiente" e in realtà non sono altro che pirati.
Il programma di supporto è identico per tutte le nazioni ma, per renderlo veramente perfetto per la realtà italiana mancherebbe una sola cosa: l'utilizzo di un canale per l'acquisto di prodotti hardware o software all'estero. Infatti uno sviluppatore italiano che cosa può sviluppare se non riesce nemmeno ad acquistare in Italia un compilatore C?

Servizi e benefici

	Commerciali	Certificati	Registrati
Abbonamento Amiga Mail	gratuito	gratuito	gratuito
Abbonamento Amiga Mail Market	gratuito	gratuito	n.d.
Sconto acquisto Hardware	(1)	(1)	n.d.
Software Public Domain	gratuito	gratuito	a pagam.
Supporto telefonico	gratuito	n.d.	n.d.
Accesso BBS	let./scrit.	let./scrit.	lettura
Sconto acquisto documentazione	40%	40%	n.d.
Documentazione ulteriore	gratuita	a pagam.	a pagam.
Documentazione hardware	gratuita	a pagam.	a pagam.
Sconto parti di ricambio	(2)	n.d.	n.d.
Beta-Test Software	gratuito	n.d.	n.d.
Beta-Test Hardware	gratuito	n.d.	n.d.
BIX amiga.com	gratuito	n.d.	n.d.
Supporto Marketing	(3)	(3)	n.d.
Sconto DevCon	da definire	da definire	n.d.

n.d. = non disponibile
(1) 25% sul prezzo praticato ai Commodore Point, IVA esclusa, pagamento contrassegno
(2) 15% sul prezzo di listino, IVA esclusa, pagamento contrassegno
(3) da valutare a seconda del progetto

Quote d'iscrizione

Sviluppatore Commerciale:	450 USD per anno, più 50 USD per apertura dossier.
Sviluppatore Certificato:	250 USD per anno, più 25 USD per apertura dossier.
Sviluppatore Registrato:	125 USD per anno, più 25 per apertura dossier.

Notizie Amiga

Hard disk 1.3, dischi RAM multipli e Video Toaster

di Don Curtis

Don Curtis è un funzionario di polizia di Denver, in Colorado. Negli ultimi due anni è stato assegnato al programma di progettazione e sviluppo e alla progettazione e manutenzione di sistema di 10 computer AT&T Unix 3B2. Durante il tempo libero Don è assistente Sysop ai CompuServe's Amiga forum.

Recentemente ho raccolto molte piccole notizie dalle reti e ho pensato di approfittare di questo numero per divulgarne il più possibile. Ho provato a usare tutti gli elementi che ho potuto, ma non ho tutte le combinazioni possibili di hardware e software, per cui non posso garantire sulla loro efficacia. Posso solo dire che le informazioni ottenute derivano da fonti attendibili, e dove possibile, sono state verificate da altre persone che le hanno provate. Tuttavia, come sempre, prima di effettuare un nuovo esperimento assicuratevi di aver una copia di backup dei vostri dati; così facendo, nel caso in cui l'esperimento non funzionasse, non avrete perso null'altro che un po' di tempo.

Ri-validazione dell'Hard Disk

Vi è capitato, cominciando a usare il nuovo Fast Filing System (FFS), di scoprire improvvisamente che avviene una rivalidazione durante un LoadWB dallo startup script?

Questo succede se avete assegnato T: alla partizione dell'Hard Disk sul vostro Old File System (OFS). Ed ecco come: l'OFS ha un bug che provoca in ogni caso un'inibizione, anche se avete attualmente un file aperto per la scrittura. Quando lanciate con il comando execute un file che avete scritto, "execute" scrive in T:; quando date il comando LoadWB, il WorkBench controlla tutta la lista di device per verificare se si tratta o meno di un drive. L'execute ha aperto la directory OFS T: per la scrittura e il WorkBench la ha appena inibita!

In questo caso, che cosa fa quindi il sistema? Rivalida la partizione OFS per definire la bit-map. Ciò è molto semplice e si ottiene assegnando T: a RAM:. In questo modo il problema è risolto.

Autoboot di una partizione FFS.

La Commodore fornisce attualmente un metodo standard per immagazzinare l'FFS nello stesso hard disk in un formato caricabile. Questo significa che, seguito il corretto procedimento, si può effettuare l'autoboot di una partizione FFS. Le periferiche della Great Valley lo possono fare mediante la loro AutoBoot ROM, come pure le HardFrame della MicroBotics. In un prossimo futuro lo farà anche la Commodore.

Coloro che hanno accesso ai file *include* della nuova versione 1.3 troveranno le informazioni in devices/hardblocks.h e resources/filesysres.h. Tali informazioni, estremamente funzionali, sono però troppo complicate per poter essere riportate qui. Vi è inoltre un enorme vantaggio: è possibile aggiornare il filesystem senza sostituire la ROM.

Problemi con l'interfaccia SCSI?

Si sono manifestati dei problemi nel funzionamento di un drive e/o una periferica SCSI? Pare che alcuni costruttori di device SCSI non curino correttamente le linee 20,22,24,28,30 e 34. Alcune di tali linee, che non dovrebbero essere collegate, vengono messe a 5 volt, mentre altre vengono collegate a terra. Qualora un device sia posto a 5 volt e la corrispondente linea di un altro device sia a terra, collegando questi due allo stesso bus SCSI si manifesteranno ovvi problemi.

La soluzione è semplice: è sufficiente non collegare queste linee al vostro cavo SCSI.

Scheda di autoboot per l'Amiga 2090

Siete infastiditi perché l'A2090 del vostro Amiga non riesce ad autoavviarsi anche se avete installato le ROM, versione 1.3, del sistema operativo nel vostro A500 o A2000? La Commodore sta venendo in vostro aiuto, non dovrete sostituire la vostra scheda A2090 con una A2090A; immetterà invece sul mercato una scheda separata per l'autoboot.

La scheda sarà composta da una ROM di autoboot che cercherà l'A2090 e, dopo averlo abilitato, lo avvierà dalla stessa. La scheda dovrà essere inserita in uno slot Amiga.

Non vi è ancora nessuna notizia relativa al costo e alla disponibilità di una tale scheda, che si chiama A2090B, in ogni caso

l'Amiga dovrà avere la versione 1.3 del sistema operativo su ROM per poter funzionare.

Anche la Michigan Software è in procinto di immettere sul mercato una scheda simile. Tale scheda sarà dotata di una RAM servita da una batteria tampone e utilizzerà una RAM statica. Avrà una capacità di memoria di 64K espandibile sino a 256K e utilizzerà 8 chip di RAM statica a 32 bit. L'Hardware non permetterà la scrittura sul disco virtuale così da impedire la perdita dei dati. Sarà in grado di abilitare tutte le partizioni e potrà autoavviarsi da un drive FFS. Avrà anche una priorità di avvio definibile dall'utente nel caso si abbia nel sistema più di un device in autoboot.

Infine avrà l'autoboot selezionabile tramite un interruttore. In questo modo, se state ancora usando la ROM 1.2, il sistema non va in crash. Un device di autoboot usato con la versione 1.2 manda in crash il sistema, ecco perché in tutti i device di autoboot attuali (come l'A2090A) le ROM non vengono installate alla spedizione. Anche in questo caso non si ha alcuna informazione su quando sarà disponibile o quanto potrà costare, si sa solo che è in via di sperimentazione.

Avvertenze per il drive SCSI della Seagate

Se avete una scheda A2090A, avrete letto le avvertenze riguardo l'uso dei drive SCSI della SeaGate (quelli che nel nome del modello hanno il suffisso N) come drive di autoboot.

Prima di tutto il problema insorge solo in un'avvio a freddo e non in una partenza a caldo. La ragione è che i drive della Seagate hanno un valore di intervallo più lungo per accendersi ed eseguire l'auto test rispetto alle schede. Se avete drive SCSI della Seagate e una scheda A2090A e volete usarli in autoboot, vi sono alcune accorgimenti da seguire.

La prima cosa da fare è avviare i drive Seagate separatamente e accenderli 15 secondi circa prima di accendere l'Amiga. La seconda è di permettere che fallisca il primo avvio a freddo e poi resettare l'Amiga usando i tasti Ctl-Amiga-Amiga. L'ultima cosa che vi resta da fare è aspettare che la Commodore esca con i nuovi drive di sua produzione sui quali stanno lavorando e che risolveranno anche questo problema.

L'uso dell'FFS in una partizione di disco MS-DOS

Volete usare l'FFS sulla partizione dell'Amiga di un hard disk MS-DOS con la Bridge-Board? Si, potete usare l'FFS con JH0: (oppure JH1:, JH2:, ecc.).

Usate la formattazione normale a basso livello e partizione del drive utilizzando FDISK e ADISK, poi, invece di usare i comandi DJMount e DJFormat per eseguire il mount e la formattazione del drive, date i comandi normali dell'AmigaDOS "mount" e "format". Poi create un elemento della mountlist per ogni partizione JHx: che avete creato. Il device si chiamerà jdisk.device (e quel file deve essere in DEVS: oppure nella linea "Device=" deve essere descritto il giusto percorso). JH0: corrisponde all'unità 0, JH1: all'unità 1 e così via. Usate i cilindri di inizio e fine appropriati quando li definite con FDISK e ADISK.

La stessa cosa per quanto riguarda il numero delle testine, delle superfici, e via dicendo. Le altre informazioni sono le stesse che si usano in ogni Fast File System drive (Buffer, FileSystem, GlobVec, DosType, ecc.). Prendete come esempio la mountlist (con le modifiche dovute) che si trova a pagina A-3 del manuale dell'1.3.

Una volta che siete sicuri di aver fatto tutto secondo le regole potete eseguire il mount e formattare il drive con i comandi standard "mount" e "format" dell'AmigaDOS. Noterete ancora un rallentamento causato dal fatto che i dati passano attraverso l'interfaccia Janus, ma avrete un accesso ai dati più veloce e un 5% in più di spazio sul drive.

L'uso di FFS con il drive Xebec 9720HD

Avete un drive Xebec 9720HD e volete usare il Fast File System? Io ho anche trovato un modo per rendere possibile questa operazione. Prima di tutto DOVETE avere la versione 3.0 dello "scsi.device" della Xebec. Potete determinare quale versione avete andando nella directory DEVS: del vostro disco di boot e dando il comando:

```
type scsi.device opt h
```

Nelle prime linee che appariranno sullo schemo ne troverete una che vi dirà il numero di versione. Se non avete la versione 3.0 NON cercate di usare il drive con un FFS, PERDERESTE dei dati.

Un altro avvertimento da dare, è che non tutti quelli che hanno cercato di fare questo hanno avuto successo. Molti ci sono riusciti ma alcuni hanno detto che sembrava che tutto funzionasse alla perfezione mentre dopo qualche giorno hanno riscontrato degli errori. Ciò è probabilmente legato a un valore sbagliato della linea MaxTransfer nella mountlist, ma non si è ancora sicuri. Per questo motivo fate copie frequenti dei vostri dati nel caso in cui vi siano dei problemi. Più avanti parleremo ancora del MaxTransfer.

Per preparare lo Xebec come drive FFS, non usate il suo comando scsimount e il suo file hdparms. Dovrete trattare lo Xebec esattamente come qualsiasi altro hard disk e creare un elemento nel file mountlist per lui usando il tipico formato FFS. Il Device dovrà essere "scsi.device" e il numero di unità dovrà essere 1. Se avete smarrito la vostra documentazione, il 9720 ha 4 testine, 612 cilindri (numerati da 0 a 611) e 17 settori per traccia. Assicuratevi ancora una volta di usare gli appropriati FileSystem, DosType e GlobVec. Come detto in precedenza, prendete come esempio la mountlist che si trova a pagina A-3 del manuale di istruzioni.

L'uso corretto di MaxTransfer

Usate nella maniera giusta le linee della mountlist di MaxTransfer? MaxTransfer è un parametro di limite, non un parametro di accellerazione!

Alcune schede di controllo dei drive possono trasferire solo una limitata quantità di dati per volta, se la vostra richiesta è maggiore la scheda di controllo andrà in sovraccarico e rovinerà i vostri dati. La soluzione a questo problema è il parametro MaxTransfer.

MaxTransfer limita la quantità di dati che possono essere letti o scritti sul drive in una singola richiesta fino al numero di byte specificati. Nella versione 1.3 del manuale c'è scritto che MaxTransfer deve essere specificato in blocchi, ma ciò non è corretto: deve essere specificato in byte.

Un modo semplice per verificare se c'è bisogno di un valore MaxTransfer nella mountlist è di creare sull'hard disk due directory: test1 e test2. Per prima cosa copiate alcuni grossi programmi (100K di dimensioni o più è sufficiente; maggiore è la dimensione del file, migliore sarà il test) nella directory test1. Quindi provate a lanciare i programmi. Se tutto funziona bene passate alla prossima fase. Se non funziona aggiungete il parametro MaxTransfer alla mountlist. Cominciate con 65536, riavviate il computer e provate di nuovo. Se tutto funziona a questo punto passate alla prossima fase, altrimenti dimezzate il valore del parametro MaxTransfer, riavviate e provate ancora.

Superata questa fase, copiate l'intero contenuto della directory test1 nella directory test2 usando l'opzione ALL del comando Copy (Copy DH0:test1 to DH0:test2 ALL). Ora provate a lanciare di nuovo i programmi. Se tutto funziona siete a cavallo! Altrimenti tornate alla procedura precedente per aggiungere il paramentro MaxTransfer, riavviate e riprovate ancora.

L'hard disk di serie

La Commodore ha annunciato che tutti i nuovi modelli Amiga saranno venduti con l'hard disk di serie. Ma non solo, l'hard disk avrà il sistema operativo già installato.

Pochi giorni dopo ha annunciato che uscirà l'Amiga 2000HD; un'Amiga 2000 con la scheda di controllo A2090A e l'hard disk da 40 MB con tempo di accesso di 28 ms, a 2999 dollari.

Hanno anche annunciato l'Amiga 2500; un'Amiga 2000 con la scheda A2090A, un hard disk da 40 MB, 28 ms, e la scheda A2620 con il microprocessore 68020 e coprocessore con 2 MB (espandibile a 4 MB) di RAM a 32 bit reali per 4699 dollari e sarà in grado di ospitare l'ambiente UNIX.

Altre notizie sulla Versione 1.3

Si presume che tutti sappiate che la versione 1.3 del sistema operativo Amiga è disponibile già da tempo e sono sorte alcune particolarità e/o correzioni interessanti.

Come abbiamo detto in precedenza, MaxTransfer è definito come il numero massimo di **blocchi** trasferiti. In effetti limita invece il numero di **byte** trasferiti. Mentre il comando "format" mostra le opzioni FFS e NOFFS, di fatto legge anche le informazioni in una mountlist, se possibile, e selezionerà correttamente il giusto metodo di formattazione (FFS o OFS) con o senza le opzioni. Andy Finkel dice che nella versione futura eliminerà quelle opzioni dal "format".

Se avete un set di chip da 1 MB (come alcuni programmatori) e usate SetPatch, il manuale dà l'informazione errata per quanto riguarda l'opzione R poiché è sensibile rispetto alle lettere maiuscole o minuscole e DEVE essere minuscola.

Drive RAM multipli

Se usate RAD: e FFS, vi sono dei problemi di ripristino del sistema. Questo si può risolvere aggiungendo la linea "Mount=1" alla vostra mountlist. Facendo ciò il sistema riesce a eseguire il recupero come dovrebbe. Un utente ha dichiarato che il recupero è avvenuto anche dopo numerosi riavvii intermedi, senza il mount del drive, sul riavvio finale; il sistema ha eseguito il mount e ha effettuato il ripristino totale.

È anche possibile avere RAD:s multiple con l'uso di un editor di settore o di file per effettuare alcune modifiche su "ramdrive.device".

Prima copiate ramdrive.device in ramdriv1.device (il nome del file DEVE avere lo stesso numero di caratteri!), poi editate il file usando un editor di file o settore (come NewZap). Nel byte 171 esadecimale troverete il carattere ASCII "e", cambiatelo in un 1. Se riuscite a vedere il file significa che avete sostituito la stringa "ramdrive.device" con la stringa "ramdriv1.device".

Ora andate al byte 5A1 esadecimale (dall'inizio del file) e cambiate il carattere ASCII 0 in 1. In questo modo avete cambiato la stringa RAMB0 in RAMB1. A questo punto duplicate la linea della mountlist di RAD: come RAD1: e cambiate la linea "Device=" in ramdriv1.device.

Eseguite il mount di RAD1:, avrete così un secondo drive RAM ripristinabile. Potete ripetere l'operazione con RAD2: e molti altri, e la memoria ve lo permetterà finché creerete per ognuno un driver e una mountlist separati con i cambiamenti necessari nel driver e nella mountlist.

Incompatibilità di software

Se nel vostro Amiga avete la ROM versione 1.3 alcuni programmi non funzioneranno. Anche in questo caso alcuni programmatori si sono riferiti alle entry codificate nell'hardware della ROM versione 1.2, che sono cambiate con la 1.3. Vorrei tanto che gli autori di programmi imparassero che nell'Amiga esiste solo UN indirizzo assoluto dal quale si può dipendere, cioè AbsExecBase. Almeno che usino quello per trovare i punti di accesso alla ROM.

Spero che le informazioni che vi ho dato vi siano utili. Come potete vedere vi sono metodi giusti e sbagliati per fare le cose, speriamo che scelgano tutti la metodologia corretta in modo che i cambiamenti apportati all'hardware o al sistema operativo non provochino più l'arresto dei programmi!

ViewPort

Il WorkBench 1.4 e oltre

di Larry Phillips

Larry Phillips è un hacker hardware e software di Amiga di Vancouver, British Columbia, ed è SysOp (System Operator) al "Compuserve's AmigaForum".

Come si poteva ben prevedere la versione 1.3 della Commodore ha fomentato varie discussioni sull'1.4, e gran parte di queste osservazioni si focalizzano su uno dei principali obiettivi dell'1.4: il miglioramento dell'ambiente WorkBench. L'Amiga è una macchina basata sulle scelte per qualsiasi cosa, dai colori e puntatori ad apparecchiature esterne hardware e, persino, per la scelta dell'interfaccia utente: il CLI o il WorkBench.

Sfortunatamente solo una delle scelte dell'interfaccia utente può essere usata per tutte le cose, mentre l'altra ha un uso limitato. Il CLI può essere usato per qualsiasi cosa, infatti il WorkBench non ha mai bisogno di essere caricato per poter usare a pieno la potenza della macchina. Il WorkBench, tuttavia, non è assolutamente adatto se comparato al CLI per qualsiasi cosa che non sia lanciare applicazioni e le più rudimentali routine di "mantenimento".

Questo stato di cose ha causato la scrittura di numerosi strumenti che cercano di dare maggior potere al WorkBench. HandyWB, Browser, IconX e altri, tutti tentano di superare i difetti dell'ambiente WorkBench. Fanno un lavoro discreto nel campo per il quale sono stati creati, ma i problemi del WorkBench sono molto più seri. Sfortunatamente le soluzioni portate avanti per un miglioramento generale del WorkBench non sempre risolvono veramente il problema, ma lo sfiorano appena.

Il WorkBench, come è adesso, è "a filo unico". Non deve essere confuso con il Single-Tasking, significa semplicemente che si deve finire una certa operazione che si sta facendo prima di poterne iniziare un'altra.

Non siete mai stati assaliti dalla noia aspettando che qualcosa si caricasse mentre osservavate il puntatore a forma di fumetto dormiente? Cliccate su una finestra, si apre immediatamente e il disco inizia a cercare le icone. Ecco che appare la prima: il programma che volete. Giunge la frustazione perché il disco deve ancora trovare cinque o sei icone e visualizzarle prima che voi possiate veramente cliccare su quella che volete. Maledite il WorkBench. "Questa è una macchina Multi-Tasking" dite, "perché non posso continuare a cliccare sulle cose?"

Vorrei che la risposta fosse semplice. Dipende tutto dal fatto che il WorkBench non sa veramente quello che volete fare. Gli state dicendo di fermare quello che sta facendo e di cominciare l'operazione seguente? Forse volete che inizi l'operazione seguente mentre continua quella precedente? Forse quello che intendete veramente è "inizia la prossima operazione quando questa è finita".

Se si tratta del primo caso potete essere sicuri che ci sarebbero molte volte in cui avreste veramente bisogno di vedere il resto delle icone, le quali naturalmente non apparirebbero mai perché il "raccoglitore di icone" è stato fermato, e naturalmente "abortire" un programma per caricarne un altro non è quello che volete fare, quindi bisognerà trattare diverse operazioni in diversi modi.

La seconda scelta, iniziare la seconda operazione mentre la prima sta continuando, è forse la cosa peggiore. E sarebbe particolarmente noioso per coloro che usano solo i floppy perché le due operazioni sarebbero in competizione per l'uso del disco.

Se non avete mai caricato due programmi contemporaneamente da un floppy provate a farlo utilizzando due diversi CLI. Litigheranno per posizionare la testina del drive e anche se alla fine riusciranno a portare a termine le loro operazioni, sarà un processo lungo e rumoroso. Tutte le lamentele sulla lentezza dell'accesso alle directory e ai programmi da floppy e il permettere l'accesso contemporaneo al disco da WorkBench fanno più male che bene all'Amiga.

L'ultima scelta, cioè il permettervi di cliccare prima che l'operazione in corso sia terminata è relativamente innocuo. La maggior parte delle volte funziona sebbene in realtà non acceleri le operazioni, ma fa in modo che l'utente abbia maggior confidenza con l'ambiente.

Un'altra possibilità che nessuno ha ancora menzionato è che l'operazione in corso venga sospesa mentre la successiva incomincia, e continui quando la successiva è terminata. Questa soluzione ha i suoi problemi, specialmente su una macchina a 512K, a meno che non si faccia attenzione a evitare l'esauri-

mento della memoria causato da una moltitudine di task in sospeso mentre ne lanciate sempre di nuovi.

Un altro argomento largamente discusso è quello della "visibilità" dei file. Un CLI vi permette di vedere tutti i file di una directory, il che è una delle principali ragioni per cui il CLI è più adatto al pieno controllo della macchina. Il WorkBench, invece, vi permette di vedere solo delle icone, e questo porta al raggiungimento di scopi ben più lontani, sia nell'utilità del WorkBench che nel modo dell'utente di avvicinarsi alla macchina.

Le icone, naturalmente, possono essere viste come rappresentazioni di oggetti. Cliccate (o cliccate due volte) su un oggetto, e aspettate che accada qualche cosa, e quello che accade dipende esattamente dal tipo di oggetto sul quale state cliccando. Questo va bene per cose come lanciare programmi o aprire cassetti, ma spesso vi sono molte operazioni che volete probabilmente eseguire utilizzando quell'oggetto e siete ostacolati dalla rappresentazione a icone. Poiché rappresenta veramente un oggetto che fa una particolare cosa, ci deve essere un altro metodo, oltre al semplice cliccare sulla icona, per indicare alla macchina che volete fare qualche cosa di diverso dalla normale operazione. In parte questo è stato fatto nella selezione dei menu su WorkBench per operazioni come duplicare (Duplicate), cambiare il nome (Rename), avere informazioni (Info), cancellare (Discard), ecc.

Il problema principale in questo caso è che l'icona, mentre rappresenta un oggetto, non necessariamente rappresenta il giusto tipo di oggetto per un altro tipo di operazione. Un buon esempio può essere il copiare un programma da un disco a un altro. Quando clicchiamo su un'icona e tenendo schiacciato il tasto del mouse la spostiamo in un altro disco o cassetto, facciamo più che non solo copiare l'icona stessa: stiamo copiando l'icona e il programma con lo stesso nome. Sfortunatamente molti programmi non si trovano in un solo file e forse neppure in un solo cassetto. In questo caso, per l'operazione di copiatura, l'oggetto è diverso. Quello che vogliamo fare effettivamente è copiare l'intero programma inclusi tutti i file di supporto.

La soluzione più comune portata avanti per risolvere questo problema è che abbiamo bisogno di vedere tutti i file su un dato disco usando il WorkBench. Alcuni dicono che dovremmo vedere le icone di quei file che le hanno, e il testo per quei file che non le hanno, mentre altri sostengono che l'immagine di default per le icone dovrebbe inserirsi nei file che non hanno una icona specifica. A prima vista questa soluzione sembra semplice e diretta, risolvendo tutti i casi, ma è più complessa.

La complessità deriva dalla filosofia della stessa interfaccia utente. Il WorkBench mira a rendere le cose più facili per l'utente e in particolare per i nuovi utenti del computer. Si suppone che elimini il timore nei riguardi del computer, nascondendo i dettagli cruenti e permettendo all'utente di vedere le cose in termini di lavori da fare, piuttosto che i dettagli di come gli stessi lavori devono essere organizzati perché funzionino.

È troppo semplice, per un utente Amiga esperto, dire che il WorkBench dovrebbe permettere di vedere tutti i file senza considerare gli effetti sul principiante di una solida barriera di icone o uno schermo pieno di nomi di file, magari che escono fuori dallo schermo richiedendo delle barre di scorrimento o altri mezzi per vederli tutti. In primo luogo questo minimizza l'utilità dell'ambiente WorkBench. Immaginatevi un utente alle prime armi che vede font, driver di device, librerie, e così via, che clicca su queste e si domanda perché non abbiano fatto qualche cosa di utile. C'è veramente differenza tra vedere tutti i files sul WorkBench e usare un CLI?

Penso che sicuramente dobbiamo essere in grado di fare un uso maggiore del WorkBench e che dobbiamo essere in grado di usarlo per vedere tutti i file, ma che questi bisogni non devono mai perdere di vista la filosofia sottostante dell'interfaccia mouse/icona. Vi sono molti modi per farlo, e alcuni di questi richiederebbero una completa rielaborazione del WorkBench, mentre altri richiederebbero il mantenimento di vecchi metodi insieme a nuovi metodi, per fare in modo che i vecchi programmi funzionino in maniera appropriata.

Il WorkBench dovrebbe probabilmente abbandonare gran parte delle caratteristiche attuali, mantenendo solo le icone, che sono molto utili in termini di lavori da effettuare, e il tipo di natura del WorkBench orientata verso l'applicazione.

Ulteriori opzioni potrebbero descrivere meglio la struttura sottostante dei file di un disco allo scopo di copiare tutte le applicazioni su un altro disco oppure per manipolare tutti i file che appartengono a un'applicazione. Un'altra opzione potrebbe dare più file, magari anche tutti, per offrire all'utente un controllo definitivo simile al CLI sulla macchina.

I proprietari di Amiga hanno proposto molte buone idee come: menù definibili dall'utente, possibilità di modificare lo sfondo, possibilità di scegliere il lato dello schermo su cui posizionare le icone dei dischi, mantenere il giusto rapporto nella raffigurazione delle icone in schermi interlacciati, creare diversi metodi per la selezione estesa, e altre ancora; molte di queste, inoltre, non hanno alcun effetto negativo sia sul nuovo utente che su quello esperto.

Alcune di queste idee possono essere considerate più che un fatto di sola estetica, e hanno la loro importanza. Tuttavia è ancora più importante che tutti gli utenti siano a proprio agio con l'interfaccia utente che più si addice a loro; il solo modo per ottenere questo è rendere gli ambienti CLI e WorkBench ugualmente potenti. Solamente permettendo delle scelte che non portino a compromessi l'Amiga potrà diventare un computer dal fascino irresistibile.

Per anni il Mac ha subito i tentativi della Apple di restringere l'attività della macchina al solo uso mouse/icona, e le macchine MS-DOS non avevano altro che la possibilità dell'interfaccia con la linea di comando. In seguito altre persone hanno fatto ogni sforzo per risolvere i difetti dell'uno e dell'altro tipo di interfaccia utente. L'Amiga è la prima macchina che ha cercato di offrire il meglio di entrambi gli ambienti pronti all'uso, e lo ha fatto in maniera egregia. Certo non è tutto perfetto, specialmente in ambiente WorkBench, e potrebbe essere migliorato considerevolmente, in modo che la filosofia di base di ogni interfaccia utente sia sempre tenuta in considerazione.

I Transputer su Amiga

Come e perché

di Howard Oakley

Il Dott. Oakley è ricercatore in fisiologia presso la Royal Navy, specializzato in sopravvivenza umana. La sua attività comprende anche la programmazione dei Transputer, di Amiga, Macintosh e PC, in modo particolare nelle applicazioni di calcolo. I suoi prodotti commerciali comprendono un programma per la progettazione delle vele navali. Attualmente sta lavorando a Mercury, un sistema operativo per Transputer. È uno dei fondatori del Transputer Users' Group.

Negli ultimi anni si è sentito parlare spesso di Transputer fra gli entusiasti di computer, anche se la maggior parte delle persone normali risulta piuttosto all'oscuro su questo argomento. Durante l'ultimo anno, in particolare, la disponibilità crescente di queste macchine, lo sviluppo dell'ATW (Atari Transputer Workstation) e le dimostrazioni tenute dalla Commodore di una scheda Transputer per Amiga, hanno creato una quantità di esperti improvvisati, di entusiasti che non vedono l'ora di poterne mettere uno nel proprio computer e di persone che sono convinte che tutti i loro problemi saranno risolti.

Sebbene i Transputer possano sicuramente offrire benefici sostanziali ai loro utilizzatori, se tutto fosse così semplice, il mondo ne sarebbe già pieno (sicuramente verrebbero prodotti in un buon numero di paesi dell'estremo est). Il progetto dell'Atari, infatti, è ormai vecchio di un anno, così come la scheda della Commodore del resto, sebbene nessuna delle due sia ancora liberamente disponibile presso i rivenditori. Chiaramente, sebbene i Transputer si prestino a risolvere certi problemi, ne devono anche creare qualcuno.

Che cos'è un Transputer

Nonostante la discussione sulla filosofia di progettazione del 68000 possa risultare di relativo interesse al programmatore di Amiga, quella del Transputer è essenziale per chiunque voglia considerare cosa farsene di uno di questi. Anziché espandere e costruire su un progetto già presente, la Inmos partì con un foglio bianco e implementò una forma di parallelismo particolarmente elegante e semplice. Questo sviluppo è stato descritto da C. A. R. Hoare in vari articoli e successivamente in un libro intitolato "Communicating Sequential Processes" e forma la filosofia di progetto sia per i chip Transputer, sia per l'*Occam*, il loro linguaggio di programmazione preferito.

Il Transputer non è, come spesso si legge, un chip RISC. Possiede invece un ingegnoso set di istruzioni che consiste di codici lunghi un byte per il 70% di quelle usate più frequentemente, che possono gradualmente diventare otto byte per istruzioni usate molto di rado.

Questo porta a un altro malinteso molto comune, riguardante i test di velocità (benchmark).

Siccome i codici delle istruzioni (opcode) di un Transputer non sono a lunghezza fissa, la velocità reale del chip è variabile. Un Transputer con clock di 20Mhz raggiungerà la velocità più alta di 10 milioni di istruzioni per secondo (MIPS) solo usando istruzioni lunghe un singolo byte. Metteteci qualche complessa elaborazione in floating point o qualche altra astrusa operazione e la velocità calerà molto presto.

Anche il set di istruzioni è abbastanza bizzarro sotto alcuni punti di vista. Il Transputer *vede* una serie di processi, ognuno dei quali possiede la propria memoria dati (zona di lavoro che può essere alterata ma non legalmente condivisa) e il proprio codice. È molto semplice preparare questi processi, in quanto è veramente solo questione di allocare uno spazio di lavoro e di metterci dentro il codice. Il processore pensa poi indipendentemente a passare il controllo fra i vari processi secondo il sistema chiamato round-robin. Per prima cosa tenta di eseguire il processo che si trova nella posizione più alta della lista dei processi pronti (ready). Se l'operazione riesce, il controllo rimane a quel processo fino a quando un certo tempo (fisso) non è passato, anche se il processore controlla il passare del tempo solo durante l'esecuzione di un piccolo numero di istruzioni fra le quali LOOP e le istruzioni per le comunicazioni. Successivamente il controllo viene passato al prossimo processo nella lista e così via.

Questo è complicato dall'esistenza di processi con priorità alta, che sono stati introdotti per permettere la programmazione di routine di tipo interrupt o trap. Un processo con priorità alta non viene interrotto dopo un intervallo di tempo, ma viene eseguito fino a quando non si deve fermare perché termina o perché deve comunicare.

Le comunicazioni avvengono attraverso dei canali, costituiti da

memoria condivisa ed efficacemente delimitata. Un processo spedisce un messaggio mettendolo nella zona del canale, mentre il ricevente lo legge prendendolo dalla locazione alla quale era stato depositato. Uno dei due può venire fermato in attesa che l'altro sia pronto a terminare le procedure di scambio del messaggio, quindi i canali possono essere utilizzati per sincronizzare più processi fra di loro, così come possono essere utilizzati per spostare dati da un processo all'altro, da cui deriva la denominazione *communicating sequential processes* (processi sequenziali intercomunicanti).

I quattro Transputer

I Transputer sono disponibili attualmente in quattro tipi diversi. Il meno pregiato è il T212, 16 bit, molto economico e potenzialmente utile nel trattamento di array molto grandi o nel controllo di periferiche. Da questo è stato derivato l'M212, progettato per il controllo dell'I/O, in modo particolare per gli hard disk ad alta velocità e per altri supporti di memorizzazione.

Il T414 è stato il primo Transputer a 32 bit. Possiede 2K di memoria interna e quattro canali di comunicazione, ma non un coprocessore (e sottrae cicli al processore durante le operazioni di comunicazione).

Il T800 è attualmente il modello di punta della serie. Possiede 4K di memoria interna, quattro canali di comunicazione (che non sottraggono cicli macchina al processore) e un coprocessore matematico molto veloce, il tutto in un solo chip. La Inmos sta già producendo limitati quantitativi di prova di nuove versioni più sofisticate della famiglia Transputer.

Le dimensioni e le maniere in cui viene utilizzata la memoria interna sono spesso critiche per le prestazioni di questi processori, in quanto sono essenzialmente privi di registri. In ogni caso, si hanno sempre a disposizione fino a 4K da riempire di codice e/o dati, i quali godono di tempi di accesso paragonabili a quelli dei registri di un chip convenzionale. Nei lavori di calcolo intensivo, per esempio, nei quali possiamo trovare poche variabili ma grosse porzioni di codice da eseguire, potrebbe risultare molto conveniente mettere le routine più critiche nella RAM interna.

Fino a un anno fa, la maggior parte degli utenti di Transputer lavoravano con il T414, poiché il T800 era disponibile solo in quantità dimostrative (e non era ancora privo di bug).

La transizione all'utilizzo del T800 non è stata facile, in quanto quest'ultimo prima di tutto è più costoso e in secondo luogo possiede un set di istruzioni diverso da quello del precedente T414. Non è nemmeno facile generare codice che possa girare indifferentemente su entrambi i modelli, dal momento che il T414 possiede alcune istruzioni non disponibili sul T800. Fino a quando il prezzo dei T800 non scenderà tanto da fare sparire i T414, ci saranno sempre questi problemi di compatibilità e la Inmos appare determinata a introdurre nuovi cambiamenti nei futuri chip.

Probabilmente l'unica caratteristica veramente innovativa di questi chip consiste nei canali di comunicazione ad alta velocità, che permettono di collegare un gran numero di CPU. La vera velocità che si ottiene da questi canali varia in modo considerevole. I T800 dovrebbero gestire 20 megabit al secondo in modo unidirezionale e poco meno durante trasmissioni bidirezionali. Queste velocità dovrebbero aumentare ulteriormente nei modelli più recenti. Ma sareste molto fortunati a ottenere prestazioni, anche solamente paragonabili a quelle di questo tipo, durante il passaggio di dati da e per un computer host (il computer che ospita la scheda Transputer), anche se si trattasse del velocissimo 68030. L'adattatore usato nel collegamento e la lentezza intrinseca dell'host ne avranno presto ragione.

Sono state proposte fondamentalmente due soluzioni per questo problema. La più semplice dal punto di vista software consiste nell'utilizzo di hardware più intelligente come la RAM a due porte, alla quale possono accedere normalmente sia il Transputer che il processore host. La più stimolante dal punto di vista software si basa invece sull'uso dell'accesso diretto alla memoria (DMA Direct Memory Access), come avviene nell'ultima scheda Transputer per PC prodotta dalla MicroWay e nel prototipo della Commodore.

Má, probabilmente, la maggiore delusione di coloro che investono denaro in costose schede Transputer e nei relativi compilatori arriva quando scoprono che i processi sequenziali intercomunicanti su un singolo Transputer sono molto diversi da quelli sparpagliati su più unità. L'approccio originale a questo problema era quello di costringere a codificare (e a eseguirne il post-link) ogni parte del software per un hardware specifico. Potevamo perciò decidere di creare il nostro programma in modo che girasse su sei Transputer, ma questo non avrebbe poi potuto essere eseguito su cinque e avrebbe sprecato quelli in eccesso se ce ne fossero stati più di sei, a meno di non modificare e ricompilare il sorgente.

La prima soluzione a questo problema venne fornita nel compilatore Parallel C della 3L e poi nei seguenti compilatori Fortran e Pascal. Questi, oltre a fornire delle routine di configurazione generale assai più flessibili, permettevano anche di riempire un qualsiasi numero di CPU con processi slave (figli) controllati da un unico processo master (padre). L'unico punto negativo consisteva nel fatto che i processi slave dovevano essere tutti uguali.

Chiaramente, per superare queste difficoltà, è necessario un sistema operativo di qualche tipo ed *Helios* è ormai maturato abbastanza da permettere di configurare arbitrariamente un numero qualsiasi di task in un numero qualsiasi di processori. Il mio tentativo di scrivere un sistema operativo, *Mercury*, è molto più semplice di Helios, ma offrirà anch'esso la possibilità di configurare i task durante il caricamento, cosicché il software possa finalmente essere relativamente indipendente dall'hardware.

L'obiettivo

L'elaborazione parallela è diventata rapidamente una delle branche più famose del mondo moderno dei computer, sebbene

le argomentazioni a suo favore siano generalmente poco comprese e portino alla ripetizione. Non appena fu sviluppato il primo calcolatore elettronico degno di questo nome, gli accademici iniziarono a speculare su quanto veloce sarebbe potuto diventare. C'è, naturalmente, un limite al flusso di dati di un processore, anche se questo limite è stato regolarmente innalzato nel corso dell'ultimo decennio.

Per ottenere le velocità di elaborazione veramente veloci di un supercomputer Cray o CDC, i trucchi per mantenere alto il flusso di dati sono diventati straordinariamente costosi: raffreddamento con liquido, semiconduttori molto costosi e così via. L'elaborazione parallela, quindi, dovrebbe essere anche una maniera molto economica di ottenere risultati paragonabili (o anche migliori). In realtà l'elaborazione sequenziale (anche su un Cray) risulta assai inefficiente per molte applicazioni che sono invece ideali per l'uso di metodi di elaborazione parallela quali le simulazioni di Monte Carlo e di altri modelli, analisi degli elementi finiti in ingegneria e così via.

I benefici ottenuti dall'inserimento dei Transputer in Amiga non sono ben definiti. Gli esempi insignificanti come quello che cita i benefici ottenuti nel calcolo della serie di Mandelbrot sono molto comuni ma di poca efficacia. Le tecniche di raytracing di Eric Graham sono mature per un tocco di parallelismo, così come lo sono i programmi seri per la grafica e il CAD. Le idee più innovative come quella dei processori di documenti basati sull'AI (Artificial Intelligence) e quella dei sistemi esperti multi-mediali potrebbero diventare obiettivi a lungo termine.

La maniera più diffusa nella quale l'utente vede i risultati prodotti dal nostro software è attraverso lo schermo, che può essere controllato da Amiga o da un Transputer. Se è l'Amiga a dovere visualizzare i risultati e la grafica, allora in molti casi finiremo davvero per utilizzare i Transputer come coprocessori, altrimenti l'overhead (il tempo aggiuntivo) richiesto dalle operazioni di comunicazione diventerebbe eccessivo. Questo approccio risulterebbe economico e potenzialmente adatto al mercato di massa.

Ad ogni modo, per portare la grafica al livello dei Transputer, permettendo a programmi completi di girare su di essi, sarebbe necessaria una scheda grafica basata sui Transputer, a costi considerevolmente più elevati. Amiga verrebbe allora ridotto a una tastiera, a poche porte di I/O molto lente e a un lento sistema di memorizzazione su disco. Dal momento che i PC stanno già ricoprendo questo ruolo nei sistemi basati su Transputer attuali, a un prezzo molto più ridotto, vedo poche possibilità di vendita in questo caso.

Qualsiasi cosa la Commodore voglia vedere fare a un Transputer in un Amiga va messo nel contesto dei loro concorrenti. La MicroWay è ormai la più grande produttrice di schede Transputer e le fabbrica solo per i PC. È in grado di offrire una serie completa di schede, inclusi sistemi grafici superbi, hard disk ultra veloci e schede I/O più specializzate. È una società ben affermata che vende ai settori corporativi e accademici, così come fanno i loro concorrenti, i quali offrono anch'essi prodotti basati su PC.

La Levco offre attualmente una serie di costose, ma ben supportate, schede per il sempre più popolare Macintosh e gode addirittura di supporto a livello di programmazione dalla stessa Apple. Atari sta chiaramente investendo molti soldi nella sua workstation, l'ATW, sebbene il suo successo a lungo termine sia messo in dubbio da alcuni.

Potrà, la scheda della Commodore, rendere superata l'ATW? Potrà competere con il massiccio mercato dei PC?

Ho la sensazione che, perché questo possa succedere, dovrà essere estremamente economica, semplice, ma fortemente supportata dai talenti di Amiga, single user e single tasking (tenete a mente che i Transputer sono molto più fragili persino del 68000, quando si usano in multitasking e sono inoltre privi di qualsiasi hardware per la gestione della memoria).

La strada attuale

Quello che è stato ufficialmente annunciato finora è che la Commodore ha prodotto una serie di quantità limitate di schede Transputer per Amiga 2000. Queste dovrebbero permettere l'uso di fino a 17 Transputer montati internamente, con la possibilità di usarne uno addizionale per la grafica (anche se questo richiede misteriosamente l'uso di uno slot PC, forse perché si tratta di una scheda per PC e non per Amiga). Il software di sistema sarà *Helios* della Perihelion, sviluppato con successo da Tim King, architetto dell'AmigaDOS e dal suo team.

Helios, dopo una partenza incerta, è oramai un prodotto maturo e abbastanza stabile. Il team ha dedicato molta attenzione a produrre un sistema operativo UNIX compatibile con pochi vizi (il peggiore dei quali è forse rappresentato dall'assenza di qualsiasi mezzo atto a intervenire in caso che il passaggio di un messaggio fallisca) e molte virtù. È abbastanza robusto nel multitasking e nel multiuser fino a potere essere impiegato nella rete di un grosso campus universitario che colleghi gruppi di Transputer sparsi negli edifici locali. Anche il server XWindows è stato al centro di un grosso lavoro per ridurne le dimensioni del codice e dei dati, per renderlo più maneggevole, almeno nei casi in cui ci siano pochi Transputer nel sistema.

In ogni caso, ci sono ancora delle piccole pecche associate a Helios. Naturalmente impone un po' di overhead sia in termini di velocità di elaborazione che in termini di memoria occupata, sebbene sia difficile fare paragoni per trarne delle percentuali precise. Originariamente sembrava molto povero nella configurazione di programmi su array differenti di processori, ma la Perihelion, per risolvere questo problema, ha prodotto un superbo linguaggio per la descrizione e la suddivisione di un processo. Comunque sia, ho paura che molti utenti non saranno semplicemente in grado di utilizzarlo, avendo bisogno di un metodo molto più intuitivo per costruire la complessità di task che formeranno i loro programmi.

La combinazione dell'hardware Commodore e del sistema operativo Helios non sembra adatta per usare i Transputer come coprocessori. Sebbene siano forniti di DMA per accelerare il trasferimento di dati, ci sarebbe bisogno anche del supporto per un uso portato al limite dei coprocessori (senza passare at-

traverso il sistema operativo) e di buone librerie di funzioni. Nonostante queste possano essere già in fase di sviluppo, non sarà molto facile che appaiano prima che Helios venga installato e ne saranno sicuramente adombrate.

Ogni volta che sento parlare di sistemi UNIX o UNIX-compatibili, divento assai poco interessato e ho paura che un numero sempre maggiore di utilizzatori di computer abbia la stessa reazione. La Apple ha imparato una lezione che la Commodore dovrebbe tenere da conto: il simil-UNIX-Macintosh , l'A/UX, è solo un aiuto nella vendita alle grosse organizzazioni quando sono già presenti tutti i collegamenti fra mini e mainframe. I Mac-entusiasti evitano A/UX e le riviste come MacTutor lo menzionano appena. Non conosco ancora nessuno che lo adoperi, nonostante abbia molti contatti con possessori di Mac II.

Pensa forse, la Commodore, che a tutt'oggi vende ancora solo PC nel mercato delle grosse organizzazioni, di avere successo là dove la Apple ha subito una dura lezione, alla faccia del NeXT di Steve Jobs, dell'ATW e della sua scatola (e di molte altre scatole prodotte da altri) con 680x0?

Chi vince?

Senza dubbio, i prodotti Commodore terranno contenta la piccola ma vocifera banda di *Amigix*, di quelli che, per una ragione o per l'altra, pensano che Amiga possa veramente evolversi per diventare una macchina UNIX. I prodotti Commodore, probabilmente, faranno anche qualche danno all'ATW Atari, anche se penso che se il prodotto di una delle due fallirà, lo faranno entrambi.

Rifuggo con orrore la prospettiva di provare a utilizzare la scheda Commodore come coprocessore, anche senza Helios e con il suo DMA. Qualcuno dovrà scrivere quelle librerie e questo qualcuno avrà più di qualche semplice problema a cui pensare. Uno dei vantaggi della RAM a due porte consiste nella possibilità di invertire l'ordine dei byte durante il processo di trasferimento, cosa abbastanza importante se dovremo trasferire insiemi misti di tipi interi e reali fra processori che vedono l'ordine dei byte esattamente in maniera opposta!

C'è poi il problema di assicurarsi che solo i pezzi giusti del codice vengano caricati nella memoria interna dei Transputer, altrimenti questi diventeranno pietosamente lenti anche in paragone a un 68030 con coprocessore 68882.

Se, quindi, state sviluppando silenziosamente una scheda Transputer che può essere prodotta in maniera economica e che calza nel ruolo di coprocessore, fatemelo sapere. Magari, dopo tutto, l'interesse principale della Commodore non sta nella commercializzazione del suo prodotto, che è già vecchio di un anno, ma nel suo uso interno per lo sviluppo e l'addestramento su un nuovo chip che sarà ancora meglio dei Transputer e che formerà la base della nuova generazione di Amiga. Nel frattempo, io ho solamente voglia di mettere il mio processore favorito nel mio microcomputer preferito.

Breakpoint

Il debugging con i programmi di Manx e Lattice

di Victor A. Wagner

Vic Wagner iniziò ad avere a che fare con i calcolatori nel 1965 quando divenne parte di un gruppo di studio della US Air Force sulla simulazione digitale del volo. Tornato un civile nel 1966, ha lavorato principalmente con costruttori di minicomputer, nel campo del software per sistemi in tempo reale. Nei giorni feriali dalle 8 alle 17, Vic cura la consulenza tecnica telefonica per la Computer Automation Inc. e la manutenzione del software di tre sistemi in tempo reale. Alla sera e nei fine settimana, trascorre il suo tempo conversando con Taarna (il suo Amiga 1000), scrivendo programmi e collegandosi ad AmigaForum. Vic è, inoltre, un'autorità riconosciuta per quanto riguarda il famoso programma di debugging MetaScope prodotto dalla Metadigm Inc.

In questi ultimi tempi un nuovo programma della Lattice, CodePRobe, si è aggiunto alla lista dei debugger che lavorano su Amiga. Non sono forse carini questi nomi che la gente sceglie abilmente per i loro prodotti? CodePRobe, con le lettere CPR maiuscole.

Siccome CodePRobe è la novità più *calda* nel campo dei debugger (beh, è comunque la più recente) ho pensato di trattarlo in questa puntata. Naturalmente parleremo anche di SDB (della Manx) in quanto questi sono gli unici due source debugger (per source debugger si intende un debugger che offre la possibilità di vedere e di operare sul codice sorgente durante il debugging) attualmente disponibili. Daremo un'occhiata agli altri debugger in una prossima puntata (ammesso che le mie dita riescano ancora a trovare la tastiera).

Si vocifera che dei source debugger siano in preparazione anche alla Avant-Garde (*Benchmark Modula-2*) e alla Metadigm (come upgrade del *Metascope*).

Sto anche tentando di scoprire che fine farà il programma LDebug (SoftCircuits) dal momento che i produttori sembrano essere andati in fallimento e dal momento che la mia copia sembra essere sparita.

Ad ogni modo, questa volta vedremo come funzionano i cosiddetti 'source' debugger e cosa rappresentano per quelli di noi che scrivono in C. Sebbene la definizione del formato dei file object preveda, fin dal primo giorno di disponibilità di Amiga, la presenza al loro interno di informazioni utili al debugging, nessuno standard è stato ancora pubblicato per stabilire come dovrebbero essere, in pratica, queste informazioni. Come risultato, si sono sviluppate diverse tecniche per fare arrivare ai debugger le informazioni necessarie.

Se me lo permettete, a questo punto vorrei inserire un commento editoriale: gran parte delle ragioni per cui Amiga sta avendo un tale successo va sicuramente attribuito allo standard IFF. Trovo che sia piuttosto deludente il fatto che, più il tempo passa, più questo affermato standard viene eroso dalla comparsa di nuovi prodotti di compagnie che *devono* fare cose speciali.

Mi spiace dover riferire che nel mondo dei source debugger le informazioni in questione non sono simili neanche nel formato. Il prodotto della Manx le scrive in un file separato, mentre quello della Lattice le seppellisce nel file eseguibile (come permesso nelle specifiche originali).

Sfortunatamente ognuna delle due compagnie sembra pensare che queste informazioni siano di carattere riservato. Naturalmente ciascuna delle due fornisce agli utenti un programma che decifra queste informazioni.

Capisco il desiderio di avere il migliore prodotto in circolazione e anche quello di avere un margine di vantaggio sulla concorrenza. Quello che *non* capisco è questo desiderio di possedere le fette di mercato come motivazione principale del proprio operato. L'introduzione di sistemi e di pacchetti software strettamente riservati condurrà al disastro che già attanaglia il mondo dei PC (*mucchi* di programmi, ma nessuno riesce a comunicare con un altro).

Bill Volk (prima alla Aegis, adesso alla Mediagenics) ha riassunto la situazione nel migliore dei modi: "non voglio una fetta più grande della torta, voglio torte più grandi". Il modo di avere torte sempre più grandi consiste nello scambio di informazioni e nella cooperazione delle compagnie che sviluppano software, anche se si tratta solamente di scambiare il formato dei dati esterni.

Ok, adesso basta, torniamo ai nostri due programmi.

Se qualcuno, nel lontano 1966, mi avesse detto (o avesse annunciato al mondo dei computer in generale) che in futuro sarebbero diventati disponibili programmi come SDB e CodePRobe, l'avremmo probabilmente rinchiuso in una camera d'isolamento.

Naturalmente, a quei tempi, una macchina *grossa* aveva 128K. Le informazioni di debugging necessarie a tenere sotto controllo un programma di medie dimensioni, specialmente un programma Amiga con tutte le sue strutture di sistema, superano da sole abbondantemente i 128K.

Okay, diamo un'occhiata agli aspetti esteriori dei debugger prima di addentrarci nei particolari del loro funzionamento.

```
cpr      169392 ---arwed 07-Nov-88 15:13:40
sdb       89228 ----rwed 24-Jan-88 01:43:48
```

Come potete vedere, CodePRobe è sostanzialmente più lungo, visto come file su disco. Una volta caricati in memoria, sulla mia macchina, le dimensioni assumono un aspetto simile:

```
CodePRobe:   size           | SDB:          size
                            |
address      Hex      Dec   | address      Hex      Dec
                            |
0296F0       224      548   | 814F10       4        4
840660       50       80    | 8428A0       214      532
842640       1680     5760  | 87A188       2058     8280
87A188       7300     29440 | 87C1E8       76D4     30420
881490       6C7C     27772 | 8838C8       2B88     11144
888118       75B4     30132 | 886458       20AC     8364
88F6D8       7B70     31600 | 888510       2330     9008
89EFD0       69D0     27088 | 88A848       3B54     15188
             ======   ======| 88E3A8       26A0     9888
total        2D0D8    184536| 890A50       2060     8288
                            | 892AB8       200C     8204
                            |              ======   ======
                            | total        1AB08    109320
```

Ricordate, e se non lo ricordate consultate l'articolo "La struttura di un programma Amiga" su questo stesso numero, che Amiga esegue il 'caricamento sparso' (scatter loading). Gli indirizzi non sono quindi di particolare interesse, ma le dimensioni degli hunk mostrano che CodePRobe è più grande anche una volta caricato in memoria.

Un'altra differenza nell'aspetto consiste nel fatto che CodePRobe viene fornito con un solenne e imponente manuale (6.5" x 9") di 138 pagine con indice e tavola dei contenuti. SDB viene fornito con un manuale (5.5" x 8.5") di 66 pagine.

Entrambi sono ben scritti e si presentano bene. Assumendo che abbiate ormai imparato come leggere un manuale, non dovreste avere alcuna difficoltà con essi.

Il manuale di SDB ha una sezione chiamata "tutorial" e quello di CodePRobe una chiamata "walkthrough". Entrambe sono utili da leggere nel caso si intenda utilizzare effettivamente il programma. SDB viene fornito con alcuni script dimostrativi che ci accompagnano in alcune sessioni di debugging a titolo di esempio.

Ragazzi, questo è uno di quei casi in cui è *altamente* raccomandabile leggere prima il manuale. Sebbene entrambi i programmi abbiano interfacce ben pensate, non posso definire nessuna delle due con l'aggettivo "intuitiva". Per favore, non prendete questo commento come critica; imparare a usare un programma così complesso, come nel caso di questi due, è come imparare un nuovo liguaggio.

Le persone che hanno scritto questi programmi usano nomi differenti per le stesse cose, così le abbreviazioni e i comandi sono alquanto diversi.

Fianco a fianco

Appena lanciamo i due programmi notiamo delle differenze. SDB ha una finestra speciale 3-in-1 che si apre a 640x200. CodePRobe apre due delle possibili quattro finestre e ne aggiusta automaticamente le dimensioni per riempire uno screen delle stesse dimensioni di quello del Workbench.

La finestra 3-in-1 può essere ridimensionata almeno fino a 680x432, nel qual caso le dimensioni delle *finestre* interne vengono aggiustate in modo proporzionale. Vi potreste chiedere perché la chiamo 3-in-1 se contiene due finestre interne ridimensionabili.

Ebbene, una delle parti è alta solo una linea ed è la parte dove appare tutto quello che digitiamo. La *finestra* al di sopra di questa linea è dove appare il codice sorgente e quella al di sotto è dove appaiono i risultati dei comandi che diamo. Il rapporto di queste due finestre si può modificare selezionando con il mouse un gadget all'estremità destra della linea di comando e spostandola in su o in giù. La linea di comando si sposta nella nuova posizione e le finestre del codice sorgente e dei risultati si espandono e si contraggono di conseguenza. Mi piace l'idea della linea di comando per l'input, in quanto non sottrae spazio nella finestra di visualizzazione per ogni comando eseguito.

Dall'altra parte, CodePRobe usa una finestra per il dialogo utente/programma abbastanza immediata. Ricorda abbastanza una finestra CLI nel senso che si vede quello che digitiamo in fondo e il contenuto della finestra scrolla verso l'alto, proprio come siamo abituati. Entrambi i programmi offrono la possibilità di richiamare i comandi già inseriti (history) e di modificarli (editing).

Una *grossa* differenza che ho notato sta nel fatto che SDB non può essere usato con un programma che non sia stato compilato in modo da avere il relativo file .dbg, mentre CodePRobe permette almeno di ottenere il disassemblato del programma e di lavorare a quel livello.

Fare il debugging con questi programmi è una vera gioia se si fa un paragone con tutti quelli che ho usato prima di passare su Amiga. Per me è semplicemente meraviglioso poter mettere un breakpoint alla linea 65 del file attuale e poi esaminare il contenuto delle variabili (perfino per nome) per poi decidere cosa

fare in seguito.

Entrambi i programmi dispongono di svariati modi per mettere i breakpoint, per associarvi espressioni condizionali (cosicché interromperanno l'esecuzione solo quando la condizione sarà soddisfatta), per eseguirli un numero ennesimo di volte prima di fermarsi e così via. Sono, insomma, in grado di scatenare le fantasie più selvagge.

Ci sono anche breakpoint temporanei e permanenti. Possiamo perfino dire al debugger di eseguire alcuni comandi quando l'esecuzione viene interrotta. I più utilizzati, in quest'ultimo caso, saranno i comandi di visualizzazione o stampa, ma le possibilità sono infinite.

Parlando di funzioni di visualizzazione e stampa, entrambi i debugger ci permettono di esaminare la memoria in qualsiasi formato possa mai essere desiderato. Per esaminare una zona di memoria qualunque, possiamo guardare i dati sotto forma di byte, word (Manx) o short (Lattice), long, puntatori, stringhe, float e la lista continua, continua, continua...

Penso che tutti questi comandi siano stati implementati per noialtri vecchi individui dalle idee arretrate che le abbiamo usate per decenni. La parte veramente eccitante viene quando il programma è stato compilato con le opzioni appropriate in modo che il debugger possa *vedere* il tipo delle variabili. Adesso tutto quello che dobbiamo fare è dire al programma di stampare una variabile e questa verrà visualizzata sullo schermo nel formato corretto.

Possiamo perfino visualizzare una struttura dati, nel qual caso tutti i suoi membri verranno stampati nel loro formato corretto, addirittura ciascuno con il suo nome. SDB menziona esplicitamente il fatto che esiste un metodo particolare per esaminare i parametri di una funzione. Sospetto che lo possa fare anche CodePRobe, ma non mi sono ancora imbattuto nella spiegazione relativa all'interno del manuale. Se avete l'impressione che io sia stato sopraffatto da questi programmi, avete ragione.

Facciamo una prova!

Diamo un'occhiata a una semplice sessione di lavoro con questi due programmi, tanto per avere un'idea di quello che è possibile fare.

Assumeremo di avere compilato ed eseguito il link del programma con successo e che quindi si sia poi lanciato il debugger. In circostanze normali questi visualizzerà il sorgente del nostro programma nella finestra superiore, con la prossima linea da eseguire evidenziata in qualche maniera.

Se vogliamo dare un'occhiata al codice generato dal compilatore (neanche io mi fido mai del mio compilatore), esiste un'opzione che permette la visualizzazione di sorgente C combinato con le relative istruzioni assembly.

Come passo successivo premiamo il tasto giusto per fare eseguire una singola istruzione dal debugger (single step). La zona evidenziata si sposta alla prossima linea da eseguire. Se stiamo guardando l'output assembly, si avanzerà di istruzione in istruzione, mentre se stiamo guardando solamente l'output C il passo sarà di una linea di sorgente alla volta. Se si continua per un certo tempo in queste operazioni, il contenuto della finestra sorgente si sposterà automaticamente per mantenere la porzione di codice sotto esame visibile sullo schermo.

Naturalmente ci si può avvantaggiare anche di una barra di scroll che entrambi i programmi mettono a disposizione. Non abbiamo neanche bisogno di fare girare il nostro editor per esaminare il sorgente, in quanto possiamo farlo comodamente all'interno del debugger. SDB mette a disposizione perfino il comando *find string* per cercare una determinata stringa all'interno del sorgente. Entrambi effettuano una numerazione delle linee del sorgente e i linguaggi di programmazione di ognuno permettono l'uso dei numeri di linea con comandi tipo *break*.

CodePRobe permette di evidenziare una linea (selezionandola due volte con il mouse), mettendo contemporaneamente un breakpoint permanente a quell'indirizzo (sebbene *non* abbia trovato alcuna informazione su tutto ciò nel manuale). Selezionando nuovamente due volte quella linea il breakpoint verrà rimosso. Sfortunatamente dopo questa operazione la finestra di comando non risulta attiva (la finestra di display diventa attiva quando clicchiamo in essa), così non possiamo scrivere semplicemente g (abbreviazione di *go*) per spostarci a quel breakpoint. Beh, questo è uno svantaggio minimo in confronto alla possibilità di selezionare una linea col mouse al fine di attivare un breakpoint in quel punto. Apparentemente SDB *non* usa il mouse per alcuna operazione (eccetto che per lo scrolling).

Entrambi i programmi hanno la capacità di togliersi di torno quando il programma sta girando. In CodePRobe, la funzione *autoswap* (così è chiamata) è normalmente ON, al contrario di SDB nel quale è OFF. Questa funzione può diventare fastidiosa, ma si può disattivare facilmente. I tasti di funzione servono in entrambi i programmi a passare dalla finestra del debugger a quella del programma in prova e viceversa.

Ci vorrà sicuramente del tempo per adattare le mie tecniche di debugging a questi due programmi. Da svariati anni a questa parte, per esempio, ho utilizzato i breakpoint dei debugger quasi esclusivamente all'ingresso delle funzioni e nelle locazioni di ritorno da esse.

La mia prima azione quando si raggiunge un breakpoint all'inizio di una funzione consiste nel trovare e visualizzare gli argomenti. Ho scoperto che entrambi i programmi hanno un singolo comando per espletare questo compito, *ds* per SDB e *where* per CodePRobe. Ma questi sono in realtà dei comandi di backtrace per lo stack. Essi visualizzano *tutte* le funzioni che sono state chiamate finora e anche tutti i relativi argomenti. Vorrei che ci fosse un semplice comando per visualizzare solamente la situazione attuale e magari anche le variabili locali.

Un altro comando che mi piacerebbe vedere implementato è "fermati dopo che questa funzione è finita". So come trovare l'indirizzo di ritorno di una funzione e come metterci un breakpoint, ma un semplice comando per fare fermare l'esecuzione del programma in quel punto sarebbe comodo. Beh, magari

non ho *veramente* utilizzato questi due debugger per un tempo abbastanza lungo da poter acquisire fluidità nel loro uso. Magari in futuro troverò un modo diverso, migliore, di fare il debugging con l'ausilio di questi due nuovi programmi.

La filosofia del debugging

C'è la tentazione, quando si usano questi programmi, di prestare meno attenzione durante la scrittura del codice (e anche durante una sessione di debugging). Sono convinto, ormai da parecchio tempo, che mettere un breakpoint in un programma sia molto simile ad affrontare un test con scelta multipla. Questo vuole dire che dovremmo già avere deciso quale, secondo noi, è la risposta giusta, prima di vedere quali sono quelle disponibili.

Ebbene, con un computer e un breakpoint si ha una sola scelta, ma l'idea è quella di decidere *prima* di premere il bottone. Sorgono sempre problemi quando si lascia che sia il nostro programma a dirci qual'è la risposta e si tenta a posteriori di decidere se il nostro programma ha ragione oppure no.

A questo punto abbiamo molti fattori che giocano contro di noi. Per prima cosa, il nostro bambino (il programma) ha appena prodotto questi risultati. Come potrebbe, una creatura così meravigliosa, avere torto? Secondariamente, è stato dimostrato ripetutamente che una figura autoritaria che produca una soluzione può fare propendere anche la persona più cinica a pensare che *lui* abbia la risposta giusta.

Se avete dei dubbi sul fatto che un computer sia una figura autoritaria, provate a recarvi in amministrazione e a spiegare che il computer *deve* avere fatto qualche errore nell'ultima busta paga.

Lasciate che vi racconti una storia

Ho visto sin troppe volte i computer intimidire la gente e convincerli a credere che *loro* avevano ragione, quando tutto il buon senso indicava diversamente. Questi computer e i loro programmi intimidiscono perfino gli stessi autori dei programmi... se li lasciamo fare.

L'esempio più clamoroso di tutto questo, che io abbia mai avuto modo di vedere, riguardava un programmatore che stava scrivendo un programma di diagnostica per un computer completamente nuovo. In questo caso il programmatore affrontava due incognite: una macchina che non aveva *mai* eseguito una qualsiasi istruzione prima d'allora e un programma che era stato appena terminato.

In sua difesa devo far presente che i nostri ingegneri accesero per la prima volta la macchina verso le 10 di sera per fare la prova iniziale con il programma di diagnostica. Era stata una giornata molto lunga e, senza ombra di dubbio, il programma di diagnostica stava segnalando che qualcosa non andava per il verso giusto nel nuovo computer. Non era una grossa sorpresa e dal momento che la macchina aveva appena eseguito la sua prima istruzione in assoluto, dovrebbe essere ovvio che questo programma non era mai stato eseguito prima.

Ad ogni modo, a mano a mano che i problemi del nuovo design (hardware) venivano localizzati e risolti, così come venivano trovati e risolti anche alcuni problemi nel programma di diagnostica, Carl iniziò ad assumere un certo tipo di approccio, che consisteva, appena un test indicava un problema, nel riportare indietro il program counter ed eseguire il test passo per passo. Sfortunatamente non si poneva a priori il problema di quale risultato il programma avrebbe dovuto produrre. Guardava i risultati mano a mano che venivano prodotti e poi diceva 'yae' o 'nae'.

Come avrete probabilmente indovinato, c'era effettivamente qualcosa di sbagliato nell'hardware della nuova macchina, e la combinazione del fare le ore piccole e del non pensare alle risposte prima che i risultati fossero disponibili cospirò nel convincere Carl che il suo programma di diagnostica era sbagliato e che la macchina stava effettivamente funzionando correttamente.

La storia ha un lieto fine. Il computer con numero seriale 0003 fu spedito a un potenziale acquirente che individuò il particolare problema quasi immediatamente. Sembra che il problema fosse abbastanza frequente nei computer nuovi e quindi l'ingegnere/il programmatore del cliente aveva effettuato un test esplicito per individuarne l'eventuale presenza.

Mi recai nella sede di questo cliente all'incirca dopo due giorni che questa macchina era stata recapitata. Era più divertito che irritato, ma fece presente che sarebbe stato meglio riparare il computer.

Ero quasi incredulo sul fatto che l'errore potesse esistere. Avevo (e ho tuttora) il più grande rispetto di Carl come programmatore (e cacciatore di bug). Non potevo credere che non avesse verificato questo tipo di bug sui confini (boundary), così chiamai l'ufficio e chiesi:

"Certo che l'ho provato, perché?" fu la risposta di Carl.

"Perché il programma di diagnostica non segnala problemi a questo proposito, ma quando lo codifichiamo a mano il test sui confini fallisce" risposi.

Carl esaminò nuovamente con attenzione il programma di diagnostica. Il test era effettivamente lì, ma il programma funzionava al *contrario*, verificava la risposta *sbagliata*. Carl e io ci facemmo sopra qualche risata e qualche birra, quando tornai in ufficio e riferii tutta la storia, ma tutta l'avventura sottolinea l'estrema importanza di fare dei test in maniera accurata.

Ci sono persone che non sono d'accordo con me nel ritenere che si dovrebbe stabilire con accuratezza quale dovrebbe essere la risposta di un certo programma, funzione, istruzione, prima di dare al calcolatore la possibilità di fornirla. La mia opinione personale è che quelli di noi che utilizzano un diverso tipo di approccio andrebbero classificati come *smanettoni* anzichè come programmatori.

Continua a pagina 61

I device di Amiga - 1

Un primo sguardo a ciò che rende unico Amiga

di Betty Clay

I device sono di importanza vitale nel funzionamento di Amiga. Il *trackdisk.device* controlla i nostri disk drive, l'*audio.device* permette ad Amiga di parlare e così via. Quando diamo il comando *Assign*, possiamo ottenere la lista di una dozzina o più device che sono attivi in quel momento nel sistema. Nel mio computer sono attualmente attivi i device Pipe:, Aux:, Speak:, Newcon:, VDK:, DF2:, DF1:, DF0:, Prt:, Par:, Ser:, Raw:, Con: e Ram:. I nomi sono familiari e il compito di ognuno di questi è conosciuto. Ma in che cosa consiste esattamente un device? E che cosa fa?

Non è mia intenzione dire in queste poche righe a un programmatore avanzato come scrivere un nuovo device per Amiga, ma chiarificarne il concetto a beneficio mio e degli utenti che ricadono nella fascia compresa fra i principianti e i medio-avanzati. I programmatori avanzati possono invece trovare sicuramente più interessante l'articolo di Steve Simpson contenuto in questo stesso numero. Inoltre, Mortimore ha riempito un intero libro di informazioni sui device e tutti quelli che desiderano implementarne uno dovrebbero prima consultarlo (*Amiga Programmer's Handbook*, volume II, Eugene P. Mortimore, Sybex).

Il termine *device* viene usato in almeno due modi differenti. Alcune volte parliamo di device come di un qualcosa facente parte dell'hardware, come un disk drive o la porta seriale, ma potremmo anche parlare di questi device fisici come di unità controllate da un device. Nell'ambiente Amiga è più accurato parlare di un device nel senso del software che controlla il device fisico. Un device Amiga controlla uno o più device fisici, ognuno dei quali rappresenta un'unità separata. Ma anche qui usiamo la parola *unit* per riferirci sia all'hardware che al software. Che confusione!

Quando ho chiesto agli esperti (ho consultato tutti i libri diponibili per Amiga), ho trovato diverse definizioni di device. Tutti erano comunque d'accordo sul fatto che device sia un concetto software e sembravano d'accordo anche sul fatto che un device sia un caso speciale di libreria di Amiga. Le librerie sono più facili da capire e sono meglio documentate, quindi risulta più facile rispondere alla prossima domanda, "Che cos'è una libreria?".

Le librerie di Amiga

Ognuna delle librerie di Amiga consiste in tre parti pricipali:

- una struttura Node, usata per collegarla al resto del sistema.
- una serie di istruzioni di salto (jump table), che danno l'esatta ubicazione delle sue svariate routine.
- un segmento di dati proprio di ogni libreria.

Per convenzione, l'indirizzo della libreria è rappresentato dall'indirizzo del primo byte della struttura nodo. I vettori della jump table si trovano a offset negativi (a indirizzi inferiori) rispetto a questo indirizzo di base, mentre i dati si trovano a offset positivi (a indirizzi superiori).

Il nodo di una libreria contiene questi elementi:

```
struct Library {
    struct Node lib_Node;
    UBYTE lib_Flags;
    UBYTE lib_pad;
    UWORD lib_NegSize;
    UWORD lib_PosSize;
    UWORD lib_Version;
    UWORD lib_Revision;
    APTR  lib_IdString;
    ULONG lib_Sum;
    UWORD lib_OpenCnt;
};
```

- una struttura Node, che abbiamo già menzionato, usata per collegare la libreria al resto del sistema
- un byte per i flag che segnalano se la libreria è valida, se il checksum è corretto ecc.
- un byte di riempimento (pad) per assicurare il giusto allineamento, non utilizzato
- una word (16 bit) contenente le dimensioni (espresse in numero negativo) della jump table

- una word per le dimensioni (espresse in numero positivo) del segmento dati

- una word per il numero di versione della libreria

- una word per il numero di revisione della libreria

- un puntatore alla stringa di identificazione, che può essere lunga fino a 255 caratteri

- una longword (32 bit) per il checksum della libreria

- una word per tenere conto del numero di task che stanno usando questa libreria

La struttura Node è la stessa usata in tutti i nodi del sistema. Qualsiasi cosa debba essere tenuta in una lista inizia con una struttura del genere. La sua composizione è molto semplice, lunga pochi byte e consistente in:

```
struct Node {
    struct Node  *ln_Succ;
    struct Node  *ln_Pred;
    UBYTE        ln_Type;
    BYTE         ln_Pri;
    char         *ln_Name;
};
```

- un puntatore al nodo successivo della lista

- un puntatore al nodo precedente della lista

- un byte che specifica di quale tipo di nodo si tratti (device, font, interrupt, library ecc.)

- un byte che definisce la priorità di questa struttura

- un puntatore al nome di questo nodo

Il byte ln_Type assicura che questo nodo venga messo nella lista di sistema appropriata quando viene caricato in memoria. Il byte ln_Pri stabilisce la posizione di questa struttura nella lista. Gli elementi con priorità più alta vengono messi in cima alla lista, assicurandogli un accesso più frequente alla CPU (nel caso si tratti di nodi task o process). Questo nodo non fa attivamente parte della libreria, ma funziona come mezzo per tenere in ordine e sotto controllo le cose nel sistema.

L'ultimo elemento della struttura Library, lib_OpenCnt, viene incrementato ogni volta che un task o un processo apre questa libreria e viene decrementato ogni volta che qualcuno la chiude. Quando questo contatore viene decrementato a zero, la libreria può essere rimossa dalla memoria e questo succederà se la macchina è a corto della stessa. Il contatore garantisce che la libreria non verrà rimossa se qualcuno la sta ancora utilizzando. Le operazioni di incremento e decremento vengono effettuate automaticamente dal sistema ogni volta che un task apre o chiude la libreria. I programmatori non hanno bisogno di preoccuparsi di nulla, eccetto che di chiudere la libreria quando hanno finito di utilizzarla.

Quando un programma chiede di aprire una libreria (con la funzione OpenLibrary), viene esaminata una lista di sistèma per vedere se la libreria richiesta è già presente in memoria. In caso affermativo, il contatore lib_OpenCnt viene incrementato e al programma viene restituito l'indirizzo del primo byte della struttura Library (che coincide con l'indirizzo del primo byte della struttura Node, in quanto essa costituisce il primo membro della struttura Library). In caso negativo, la libreria viene caricata in memoria, inserita nella lista di sistema, il contatore viene incrementato e, come prima, ne viene restituito l'indirizzo di base al programma invocante.

L'ubicazione di una libreria contenuta nel Kickstart può essere differente per ogni versione del sistema operativo. Nel caso di una libreria che si trova su disco (nella directory LIBS:), l'ubicazione in memoria cambia ogni volta che la libreria viene caricata, ma la posizione della jump table all'interno della libreria rimarrà sempre costante. Questo è il modo che permette al software di rimanere compatibile con macchine diverse o con versioni diverse del software di sistema. Se il programmatore chiama una routine di una libreria seguendo il metodo standard, il sistema sarà sempre in grado di trovarla, non importa dove si trovi in quel momento.

Ogni funzione avrà un offset negativo dall'indirizzo di base della libreria e i dati ne avranno uno positivo (da qui il motivo delle dimensioni negative e positive lib_NegSize, lib_PosSize). Ogni elemento della jump table è contenuto in sei byte: due per l'istruzione di salto (JMP) e quattro per l'indirizzo della routine. Le routine vere e proprie possono essere posizionate in qualsiasi parte della memoria, ma i loro indirizzi verranno immagazzinati nella jump table e da questa il task potrà rintracciare le istruzioni che gli servono.

Come fa un programma a sapere dove trovare una determinata funzione, quando il sistema operativo gli fornisce solo l'indirizzo della libreria?

Qui entrano in gioco il compilatore e il linker. Quando si esegue il link di un programma, il linker individua l'offset all'interno della jump table per ogni routine che il programma chiama e codifica questo offset nel programma. Durante l'esecuzione il programma si porta all'indirizzo base della libreria, quindi si sposta dell'offset fornito dal linker e salta all'indirizzo della vera routine.

La parte contenente dati di una libreria può essere consultata da un task, ma normalmente contiene informazioni pertinenti solo alle routine interne della libreria stessa. In alcuni casi i corpi delle routine della libreria sono contenuti in questa sezione. Questa parte della libreria non è così ben descritta e documentata come il nodo e la jump table discussi in precedenza.

I device

Uno degli obiettivi dei progettisti di Amiga era di costruire ogni parte del sistema nella maniera più coerente possibile. Questo avrebbe significato facilità di programmazione. Nel caso dei device, comunque, questo si è rivelato estremamente difficile. Una penna ottica non richiede, per esempio, gli stessi

comandi necessari per utilizzare un disk drive. Il timer.device ha bisogno di segnali di tipo diverso da quelli richiesti dall'audio.device. I device, quindi, sono stati codificati nella maniera più coerente possibile, ma ognuno di essi richiede routine specializzate per lo svolgimento dei propri compiti e di cui è difficile fare una trattazione generale.

Siccome un device è derivato da una libreria, sappiamo che deve contenere un nodo come nelle librerie, una jump table per le funzioni del device e una struttura dati, e che queste devono seguire le regole già viste per le librerie nel senso di avere degli offset negativi per la jump table e positivi per i dati. Ecco infatti come è definita la struttura device:

```
struct Device {
  struct Library dd_Library;
};
```

Nel *Rom Kernel Manual* troverete una descrizione dettagliata di ciascun device, nella quale vengono analizzate le sue funzioni caratteristiche. Ma esiste anche una serie di funzioni standard presenti in tutti.

La jump table di ogni device conterrà le chiamate alle routine standard di ogni libreria *Open*, *Close*, *Expunge* e la funzione a uso riservato *ExtFunct*. Ci saranno anche i vettori di *BeginIO* e *AbortIO*. La routine BeginIO viene chiamata passandogli come argomento il puntatore a una struttura IORequest, nella quale il programmatore ha memorizzato informazioni sul device che deve essere utilizzato e i comandi che questi ha bisogno per funzionare. Questi ultimi potrebbero essere comandi standard come Open, Read, Write o Close, oppure potrebbero essere dei comandi caratteristici del device in questione come il comando per allocare un canale audio per l'audio.device. La maggior parte dei device conterrà nella jump table anche i vettori per una serie di routine che controllano le caratteristiche particolari dell'hardware legato al device in questione.

Ogni device è collegato a delle strutture chiamate *Unit* (unità).

```
struct Unit {
  struct MsgPort   unit_MsgPort;
  UBYTE unit_flags;
  UBYTE unit_pad;
  UWORD unit_OpenCnt;
};
```

Il *trackdisk.device* costituisce un esempio lampante, dal momento che possiamo avere più drive controllati da un unico device. In questo caso ci sarà una struttura unità per ciascuno dei drive utilizzati. Questa unità, comunque, non è un concetto hardware. È una struttura software che permette di passare messaggi e dati da e per il task o processo che accede all'unità logica. Ogni device dispone di almeno una struttura unit associata con esso.

Dal momento che il compito principale di ogni device consiste nel gestire l'input e l'output di dati, è necessaria una buona comprensione dell'I/O per capirne il funzionamento.

Quando un task apre un device chiamando OpenDevice, il sistema operativo inizializza automaticamente una struttura Device e anche una struttura Unit, il cui membro unit_MsgPort servirà come message port per la particolare unità selezionata. Lo stesso device e la stessa unità possono essere condivisi da qualsiasi altro task in esecuzione e il membro lib_OpenCnt della struttura Device (lo stesso della struttura Library) terrà conto del numero di task che li stanno usando.

Un task userà una struttura di tipo IORequest per descrivere le sue necessità di dati e manderà questa richiesta alla porta della struttura unità. Il messaggio dirà al device quali dati vengono richiesti e dove metterli in memoria. Normalmente queste richieste vengono soddisfatte sequenzialmente, chi arriva prima viene servito prima, ma in alcuni casi il task richiedente può specificare che la richiesta sia di tipo QuickIO, facendole saltare la fila e ottenendo una risposta il più presto possibile. Il device, comunque, è abbastanza intelligente. Se gli risulta impossibile soddisfare questo tipo di richiesta immediata in un tempo ragionevolmente breve, la richesta non verrà respinta, ma verrà messa in coda come tutte le altre e processata quando arriverà il suo turno.

Tutto questo viene gestito automaticamente dal software di sistema e il programmatore deve solo preoccuparsi di preparare e spedire l'appropriata struttura di tipo IORequest.

Quando un task ha finito di lavorare con i dati contenuti nel suo buffer in memoria e spedisce la richiesta che questi dati vengano salvati su disco, questi verranno trasferiti dal buffer del task al buffer del device e attraverso quest'ultimo verranno spediti direttamente all'hardware.

Perché?

Le librerie e i device fanno un ottimo uso della memoria. Il codice può essere posizionato ovunque nella memoria e le liste di sistema manterranno il tutto unito e disponibile. In questo modo è facile aggiornare il sistema operativo, perché, dal momento che le librerie e i device sono gestiti dalle jump table, modificare i vettori in queste ultime per puntare a nuove routine non crea alcun problema al software applicativo.

L'ordine dei vettori in queste tabelle rimane costante nel tempo e un vettore specifico viene rintracciato tramite indirizzamento indiretto. Il sistema rintraccia una routine grazie al suo nome, utilizzando uno dei linguaggi ad alto livello, in modo che non sia più necessario memorizzare liste di locazioni come succedeva nella vecchie macchine a indirizzamento assoluto.

L'uso di librerie e device è particolarmente importante in un computer multitasking. Una volta che uno di questi è stato caricato in memoria, può essere utilizzato da tutti i task in esecuzione. Ogni singolo programma può avere delle dimensioni assai ridotte perché non ha bisogno di contenere tutte le funzioni, che gli vengono invece messe a disposizione dalla libreria o dal device. E infine, è molto più semplice scrivere un programma quando queste routine di frequente utilizzo sono già state scritte per noi.

Amiga Structure Browser

Visita guidata attraverso il sistema

di Chris Zamara e Nick Sullivan

Con l'aiuto di questo programma e delle informazioni contenute in questo articolo riusciremo a ottenere delle informazioni preziose sul funzionamento di Amiga che non riuscivamo a trovare sui manuali. Questa conoscenza ci permetterà di migliorare la nostra attività di programmazione e di renderla quasi sicuramente più proficua di quanto lo sarebbe stata altrimenti.

Con tutti i soldi che guadagnerete da questo incremento produttivo, potrete comperare l'hardware più costoso e quindi migliorare ulteriormente le vostre capacità. Questo vi renderà programmatori di successo e questo successo aumenterà la confidenza in voi stessi. La gente si rivolgerà a voi per avere consigli e così aiuterete molti altri che vorrebbero essere così bravi e così di successo come voi. Nell'aiutare queste persone, non solo ne ricaverete una personale profonda soddisfazione, ma diventerete anche famosi e benvoluti, cosa che certo non danneggerà la vostra vita sessuale. Dal momento che così tante persone vi prenderanno come punto di riferimento e seguiranno i vostri consigli, vi troverete a poter disporre di un considerevole potere nell'industria dei computer e nelle vostre contrattazioni di affari in generale. Nel sentirvi soddisfatti di voi stessi, godrete sicuramente di una forma fisica migliore e vivrete sicuramente più a lungo, magari fino a più di trecento anni!

In breve, leggere questo articolo e usare il programma relativo vi porterà sapienza, potere, successo e prosperità. Migliorerà la vostra vita sessuale e vi darà una maggiore stima di voi stessi e la sensazione di essere utili. Potrete comperare i computer più costosi e le macchine più veloci, se lo desiderate. Vivrete più a lungo. Ma la scelta è vostra: leggere questo articolo e realizzare questi cambiamenti nella vostra vita o saltare più avanti e rinunciare a realizzare gli enormi potenziali che sapete di possedere.

Beh, magari non otterrete tutte queste cose, ma se volete saperne di più sulle strutture interne di Amiga, o se volete un mezzo facile per arrivare ai dati vitali di un programma mentre questo è in esecuzione, farete buon uso dello Structure Browser di Transactor.

Sappiamo bene che le varie strutture di sistema sono molto importanti, dal momento che rappresentano la chiave per accedere a tutte le entità conosciute dal sistema, come i task, gli screen, le window, i gadget e tutto il resto. Su Amiga non esistono locazioni di memoria fisse, quindi le strutture sono l'equivalente della mappa di memoria che trovavamo sulle macchine non multitasking.

La struttura è un costrutto del linguaggio C, ma possiamo parlare della definizione di una struttura dati indipendentemente da qualsiasi linguaggio. Una struttura è semplicemente un modo particolare usato per raggruppare alcuni dati. Una struttura chiamata 'X', per esempio, potrebbe essere definita come contenente una word (16 bit), seguita da un puntatore a un'altra struttura 'X' (32 bit), seguito da un puntatore alla struttura 'Y' (32 bit), seguito da una longword (32 bit). Conoscendo la definizione di una struttura e la sua posizione in memoria è molto semplice per il sistema operativo o per il programma applicativo raggiungere le informazioni desiderate.

Che tipo di informazioni possiamo trovare nelle varie strutture del sistema operativo? Ebbene, si possono scoprire alcune cose interessanti, per esempio, nelle strutture controllate da Intuition. Con il programma SB saremo in grado di esaminare, fra le altre, le strutture Screen, Window e Gadget. Da queste si possono ricavare informazioni su qualsiasi altro programma attivo nel sistema, in quanto gli screen, le window e i gadget di ognuno sono collegati come in una specie di rete, dove possono essere raggiunti attraverso l'utilizzo di puntatori contenuti nelle altre strutture Screen, Window e Gadget.

Un puntatore alla prima struttura Screen è tutto quello che serve per trovare tutto il resto. Questo puntatore può essere rintracciato nella struttura "IntuitionBase", che, a sua volta, viene fornita quando la libreria di Intuition viene aperta in seguito alla chiamata della funzione OpenLibrary() (la funzione OpenLibrary() è contenuta nella libreria dell'Exec, il cui puntatore si trova alla locazione assoluta 000004, l'unica locazione fissa nell'intera mappa di memoria di Amiga. Il codice di startup di un programma C apre automaticamente la libreria dell'Exec). Tutti i programmi che usano Intuition memorizzano le loro informazioni in queste strutture, così attraverso di esse possiamo apprendere i particolari di praticamente tutti i programmi che sono attualmente nel sistema.

Una struttura Screen contiene tutto quello che Intuition ha bisogno di sapere di uno screen, come le sue dimensioni, il titolo, il numero di bit plane, il puntatore ai gadget associati a questo screen, i flag che descrivono di che tipo di screen si tratti, i colori nei quali deve essere visualizzato, ecc. La struttura Screen contiene anche un puntatore alla struttura Window che appartiene alla prima window aperta su quello screen. Una struttura Window contiene un puntatore che può indirizzare un'altra struttura Window, così tutte le window di uno screen risultano collegate insieme. Se ci sono più screen presenti nel sistema (se non sta girando un programma applicativo che apre il proprio schermo, l'unico screen normalmente presente è quello aperto dal Workbench), ci sarà un puntatore, contenuto nella prima struttura Screen, che indirizzerà la struttura Screen successiva e così via. In questa maniera possono essere collegati insieme un qualsiasi numero di screen.

In una struttura Window troveremo le solite informazioni riguardanti le dimensioni e il colore, insieme alla variabile IDCMPFlags che descrive quali messaggi Intuition sta ricevendo da quella finestra. La struttura Window contiene anche un puntatore al primo di una serie di gadget. Una struttura Gadget descrive completamente un gadget (un mezzo estremamente semplice per ottenere un input dall'utente per mezzo del mouse ed eventualmente della tastiera), la sua posizione, le dimensioni, il tipo, la grafica, ecc. La struttura Gadget viene preparata dal programma applicativo prima di essere consegnata a Intuition, così i valori in essa contenuti rappresentano quelli che il programmatore ha inserito nel suo codice. In parole semplici, dare un'occhiata alle strutture Gadget di un programma può fornire qualche indizio su come è stato scritto il programma stesso.

Altre strutture estremamente interessanti, che però non sono state ancora implementate nella versone 1.0 di Structure Browser sono: Menu, Image, Requester, View, ViewPort, Layer, Interrupt, MemChunk, Message, Task, VSprite, AnimOb (molte di queste strutture, come la Task e la Interrupt, sono difficili da osservare, poiché i dati che contengono cambiano continuamente).

La chiave per trovare la locazione in memoria di qualsiasi struttura è la struttura Library. Qualsiasi libreria di sistema ("graphics.library", "intuition.library", "layers.library" per nominarne solo alcune) possiede una tale struttura, il cui indirizzo è conosciuto come 'base della libreria', che al suo interno contiene puntatori alle varie strutture che riguardano le funzioni di quella libreria. Come abbiamo detto precedentemente, le strutture formano una specie di rete, poiché possiamo raggiungere qualsiasi struttura attraverso puntatori contenuti in altre strutture. In questa rete possono essere rintracciati praticamente tutti i dati che risultano di una qualche importanza ai fini del funzionamento del sistema operativo. In sostanza possiamo ricavare uno spaccato dello stato generale del sistema in un qualsiasi momento. Se conosciamo le definizioni delle strutture di sistema (ce ne sono più di cento) possiamo vagare e scoprire qualsiasi cosa che il sistema conosce e in Amiga il sistema operativo è molto informato, perché il software applicativo deve passare attraverso il sistema per quasi qualsiasi cosa intenda fare.

Il programma Structure Browser

Il programma SB rende facile andare a curiosare attraverso molte delle strutture di sistema. Si comincia con una lista delle librerie disponibili. Nella versione attuale è stata implementata solo quella di Intuition. Per visualizzare il contenuto della sua struttura Library, basta muovere il mouse sul nome e premere il tasto sinistro. La struttura della libreria di Intuition, chiamata IntuitionBase, verrà immediatamente visualizzata, completa di tutti i nomi dei suoi membri (così come sono definiti nei file header standard) e del loro tipo. Nel caso di IntuitionBase, i membri sono costituiti da strutture e da puntatori a strutture. Per investigare ulteriormente su una qualsiasi di queste strutture è sufficiente usare il mouse come spiegato prima. Il procedimento può essere ripetuto nella nuova struttura e così via, fino a quando ce ne sia almeno un'altra disponibile.

Si può sempre ritornare nella struttura dalla quale proveniamo, selezionando il gadget "torna indietro", situato nell'angolo in basso a sinistra della window. Si possono inseguire i propri passi a ritroso come si desidera, dal momento che il programma lavora in maniera ricorsiva (il livello di profondità raggiunto è visualizzato nell'intestazione della finestra). Le strutture che contengono più membri di quanti la finestra del programma possa ospitare verranno divise in pagine di 16 linee e apparirà il gadget "ancora". Possiamo spostarci, quindi, alla pagina successiva usando il gadget "ancora" e alla pagina precedente usando il gadget "pagina preced".

Nell'andare di struttura in struttura è facile apprendere alcune informazioni sui programmi che non si sarebbero potute conoscere altrimenti. Per esempio, che tipo di gadget vengono utilizzati in quel bel programma commerciale? Sono veramente dei gadget? Il programmatore ha usato un requester o una finestra per visualizzare quel messaggio? Con il semplice uso del mouse e di SB possiamo scoprire questo e altro!

La classe (il tipo di dato) di ogni membro di una struttura viene visualizzata traendola direttamente dalla sua dichiarazione nei file header del C (il file header appropriato deve essere compilato, utilizzando il comando #include, insieme a ogni programma che intenda accedere a una certa struttura di sistema). Queste classi sono standard su Amiga, in quanto vengono dichiarate utilizzando il comando typedef all'interno del file "exec/ types.h". I tipi fondamentali sono:

```
BYTE   - 8 bit con segno (char)
UBYTE  - 8 bit senza segno (unsigned char)
SHORT  - 16 bit con segno (short)
USHORT - 16 bit senza segno (unsigned short)
LONG   - 32 bit con segno (long)
ULONG  - 32 bit senza segno (unsigned long)
```

I puntatori ai tipi sopraelencati sono contraddistinti, seguendo la sintassi standard del C, con un asterisco (*). Per esempio, un puntatore al tipo SHORT viene scritto SHORT *. La sintassi del C viene seguita anche per i membri che rappresentano puntatori ad altre strutture. Il puntatore a una struttura di tipo Window, per esempio, viene denotato da 'struct Window *'. Se un membro di una struttura è a sua volta una struttura, non un

puntatore a essa, viene denotato senza asterisco. SB, in questo caso, non stampa alcun indirizzo per tale membro, dal momento che è contenuto all'interno della struttura attualmente sotto esame, della quale si conosce già l'indirizzo.

I membri di una struttura che non siano puntatori ad altre strutture ma dati reali che possono risultare interessanti, vengono visualizzati in modi differenti, a seconda della loro natura. Questi dati possono essere semplici numeri, come quelli che forniscono i parametri LeftEdge, TopEdge, Width e Height di una finestra. L'elemento interessante potrebbe essere un byte, una word o una longword contenente dei flag di qualche sorta, come quelli indicanti il tipo di una finestra o di un gadget. Oppure, i dati che suscitano il nostro interesse potrebbero semplicemente consistere in un'area di memoria contenete una tabella di qualche genere o descrivente un'immagine grafica.

I semplici dati, come LeftEdge ecc., sono stampati in nero nella finestra di SB. Questi membri non possono essere usati per accedere ad alcuna informazione supplementare, quindi nulla succederà se tentiamo di selezionarli con il mouse. Essi sono semplicemente quello che sembrano: un numero e basta. I valori stampati in nero rappresentano spesso proprio quello che volevamo sapere di una certa struttura. Quando uno di questi valori viene utilizzato per memorizzare dei flag, viene trattato diversamente e quindi viene stampato in bianco. Possiamo infatti selezionarlo per avere l'elenco dei flag attivati. Un esempio può essere costituito da una window di tipo "smart refresh", fornita di sizing, close e drag gadget. Il membro "Flags" di una tale struttura Window, quando selezionato, visualizzerà:

```
WINDOWDRAG      WINDOWSIZING
WINDOWCLOSE     SMART_REFRESH
```

(come i nomi dei membri di una struttura, anche i nomi dei flag sono definiti nei file header standard usati per lo sviluppo di programmi C o Assembly).

Un'altra forma di output particolare di SB è la stampa esadecimale. Quando il membro di una struttura è un array di valori o il puntatore a un'area di memoria rappresentante un bit-plane o i dati di un'immagine, possiamo ottenere una stampa esadecimale selezionandone il nome come al solito. Il tipo di dati in questione, se presente nella definizione, viene tenuto in considerazione durante la stampa e quindi i dati vengono suddivisi in gruppi di byte, word o longword per la visualizzazione.

Un giro di prova

Sulla carta il funzionamento di SB potrebbe suonare abbastanza complesso, ma in realtà il suo uso è molto semplice. Vediamo un esempio di come potere scoprire il tipo dei gadget utilizzati da un programma attualmente in esecuzione. Proviamo a usare come esempio lo stesso programma Structure Browser. Cercheremo di identificare la costituzione del gadget "torna indietro".

Per prima cosa lanciamo SB. Questo non è difficile: basta digitare 'sb' nel CLI (o 'run sb' se vogliamo potere disporre ancora del CLI mentre SB sta girando). Apparirà la finestra di SB, leggermente spostata verso il basso, in modo da lasciare visibile un pezzetto dello screen del Workbench. Ci verrà chiesto di selezionare la struttura di una libreria. Dal momento che in questa versione del programma ne è disponibile una sola, la scelta non risulta difficile. Selezioniamo "Intuition" puntandola con il mouse e premendo il tasto sinistro. Dopo questa operazione, verrà visualizzato il contenuto della struttura "IntuitionBase", ogni membro con il relativo tipo e valore visualizzati sulla destra. Tutti i membri consistono di una struttura o di un puntatore a essa. I primi due membri sono messi fra parentesi per segnalare il fatto che non sono ancora disponibili per un ulteriore approfondimento in questa versione del programma. Allo scopo di mantenere le dimensioni del listato di SB ragionevoli per la pubblicazione su queste pagine, abbiamo implementato solo alcune fra le strutture più importanti. I riferimenti a tipi di dati non implementati vengono messi fra parentesi e la loro selezione non produce alcun effetto.

Continuiamo nel nostro viaggio per raggiungere il gadget in questione. Siccome il programma al quale siamo interessati ha una window sullo schermo del Workbench, dobbiamo prima raggiungere la struttura di quest'ultimo. Dal momento che lo schermo attivo in questo momento, quello nel quale stiamo lavorando, è proprio quello del Workbench, possiamo recarci lì selezionando il membro "ActiveScreen" della struttura IntuitionBase. Basta la pressione del tasto sinistro del mouse ed ecco apparire la prima pagina di informazioni sulla struttura Screen del Workbench. Possiamo verificare di avere raggiunto lo screen desiderato esaminando il membro chiamato "Title". Alla sua destra dovrebbe comparire "Workbench Screen". Per vedere gli altri membri della struttura Screen basta selezionare il gadget "ancora" che è apparso nella zona bassa della finestra. Il gadget "pagina preced" ci farà tornare alla pagina precedente. Il secondo membro della struttura Screen si chiama "FirstWindow" e punta alla prima struttura Window nella lista che contiene tutte le window di questo screen.

Selezionando "FirstWindow" verrà visualizzata una struttura Window. Molto probabilmente questa è la finestra che stavamo cercando, la finestra di SB. Il membro "Title" ci dirà quale finestra stiamo esaminando perché riporta fedelmente quello che appare nella barra riservata al titolo della medesima. Se la finestra non è quella che cercavamo, possiamo esaminare quella successiva selezionando il primo membro della struttura, chiamato "NextWindow". Verrà visualizzata un'altra struttura Window, appartenente alla finestra successiva nella catena. Fermiamoci quando raggiungiamo la finestra di Structure Browser (lo possiamo verificare da "Title"). Possiamo notare altri elementi interessanti in questa struttura, quali le dimensioni e i flag. Proviamo a selezionare il membro "Flags" per vederne il contenuto in forma esplicita. Anche se non siamo molto familiari con la programmazione di Intuition, i nomi dei flag ci possono suggerire il loro significato.

Adesso che abbiamo trovato la struttura Window che cercavamo, vediamo di trovare il puntatore alla lista di gadget relativa a questa finestra. Non c'è nulla del genere nella prima pagina, quindi selezioniamo il gadget "ancora" per vedere gli altri membri. All'incirca a metà della seconda pagina troviamo il

membro chiamato "FirstGadget" il quale punta a una struttura Gadget (il suo tipo è 'struct Gadget *', che significa 'puntatore a struttura Gadget'). Selezioniamolo, ed ecco apparire la prima di una serie di strutture Gadget. Esaminando le variabili LeftEdge e TopEdge possiamo risalire alla posizione del gadget nella finestra. Per raggiungere il prossimo gadget nella lista è sufficiente selezionare il primo membro della struttura, chiamato "NextGadget". Possiamo scorrere tutta la lista dei gadget in questo modo, fermandoci quando vogliamo esaminare le variabili "Flags", "Activation" o "GadgetType". Quest'ultima ci rivelerà se il gadget sia di tipo BOOLGADGET (booleano), PROPGADGET (proporzionale) o STRGADGET (stringa).

Prima o poi raggiungeremo un gadget con TopEdge di -12 e LeftEdge di 10. Dal momento che il flag GRELBOTTOM è settato nella variabile "Flags", sappiamo che il valore TopEdge indica che il bordo superiore del gadget dista di 12 pixel dal bordo inferiore della finestra. Questo allora è il gadget "torna indietro" che stavamo cercando e ciò ci viene confermato esaminando il membro "GadgetText" della struttura gadget. Possiamo vedere esattamente come è stata preparata questa struttura Gadget nel programma Structure Browser (usare gli stessi dati dell'immagine di un gadget contenuto in un altro programma per crearne uno analogo all'interno di un nostro programma, potrebbe essere considerato una violazione del copyright. La versione attuale di SB non supporta l'esplorazione delle strutture "Image", quindi è inutile fare qui raccomandazioni del tipo "Non fatelo a casa, bambini!").

Il programma SB non supporta tutte le strutture di sistema, altrimenti riempirebbe l'intera rivista. A beneficio di coloro che ordineranno il disco di Transactor, abbiamo accluso, oltre alla versione 1.0 pubblicata in queste pagine, la versione 1.3 che ne incorpora qualcun'altra. Il programma, comunque, è progettato in modo che risulti abbastanza facile aggiungere le strutture di nostro gradimento, partendo dai file "include" che si trovano sul disco di sviluppo del C e sul ROM Kernel Manual. Tratteremo di questo più approfonditamente nel seguito dell'articolo. SB è un programma di pubblico dominio, quindi siete liberi di usarlo e modificarlo a piacere.

Applicazioni di SB

A questo punto vi sarete resi conto che SB rappresenta un grande aiuto per apprendere la costituzione delle strutture interne di Amiga. Ma ci sono alcune altre buone ragioni per tenere SB nei dischi che usiamo maggiormente, magari nella directory C del disco Workbench.

Per prima cosa, SB costituisce un comodo mezzo per ottenere la definizione di una struttura. È più facile lanciare SB e selezionare una struttura per averne la definizione, piuttosto che andare a sfogliare il ROM Kernel Manual. E se non possediamo questo manuale, è sicuramente più facile che cercare la definizione in mezzo a così tanti file include. Se poi non abbiamo neanche questi ultimi, allora è proprio l'unico mezzo a disposizione!

SB può essere usato anche come strumento di debugging. Se il nostro programma si comporta stranamente, possiamo dare un'occhiata a come Intuition vede le nostre finestre, i nostri gadget, per essere sicuri che siano tutti correttamente inizializzati e collegati l'uno con l'altro, proprio come intendevamo che fossero. Possiamo fare un giro esplorativo in qualsiasi momento dell'esecuzione del nostro programma, per vedere quali sono gli effetti prodotti da una certa azione intrapresa. Una specie di strumento di trace, insomma, ma che riguarda solo l'interazione del programma con il sistema operativo.

Un altro beneficio offerto da SB consiste nel potere imparare le tecniche di programmazione utilizzate dagli altri programmi, nel preparare le loro strutture, per ottenere un determinato effetto. Nell'investigare in merito a entità conosciute del sistema operativo, possiamo vedere quali effetti hanno sul loro comportamento il cambiamento di certi flag e di certe variabili. Se siamo curiosi di conoscere quali i trucchi stia utilizzando un programma che abbiamo appena lanciato, basta dare un'occhiata alle sue strutture. Per esempio, lo sapevate che il Workbench utilizza una window BORDERLESS e BACKDROP che si trova sullo schermo del Workbench appena sotto la barra che contiene il nome dello schermo? Noi non lo sapevamo, fino a quando non lo abbiamo scoperto curiosando con SB. Si possono trarre delle buone idee esaminando l'uso che gli altri fanno delle risorse messe a disposizione da Intuition.

Versioni future di SB

Il programma può essere espanso a piacere e può crescere notevolmente di dimensioni, ma ne varrebbe la pena per avere una visione completa del funzionamento interno di Amiga. Oltre all'implementazione di tutte le strutture mancanti nella versione 1.0 e nella 1.3, si potrebbe migliorare notevolmente il modo in cui vengono gestiti i dati di basso livello. I dati grafici, per esempio, potrebbero essere trasformati nelle relative immagini, anziché visualizzarli come stampa esadecimale. Alcuni valori, come le coordinate del mouse che si trovano nella struttura Window (MouseY e MouseX) potrebbero essere aggiornate continuamente, per darne una lettura in tempo reale. Le stampe esadecimali potrebbero essere affiancate dall'equivalente ASCII. I dati delle immagini e dei BitMap dovrebbero poter essere salvati su disco in formato IFF. Potrebbe esserci un'opzione per salvare su disco il contenuto di una qualsiasi struttura in linguaggio C o Assembly, per poterla poi riutilizzare nei nostri programmi e via di questo passo. Se qualcuno ha qualche idea valida, la implementi! Non vogliamo tenerci tutto il merito e la gloria! (leggi: non vogliamo fare tutto il lavoro da soli!)

Note al listato

Il sorgente di SB è diviso in vari file, ognuno con uno scopo ben specifico. Il cuore del programma e le sue funzioni principali sono contenute nel file "sb.c". Le chiamate a Intuition e tutte le funzioni che gestiscono l'interfaccia con l'utente si trovano nel file "sbio.c". Gli altri file contengono le routine che gestiscono i vari tipi di strutture. I file "sbWindow.c", "sbScreen.c", "sbGadget.c" e "sbNode.c" contengono le routine per gestire le strutture implicate nel loro nome. "sbGraphics.c" gestisce le strutture RastPort e BitMap relative alla "graphics.library". Il file "sb.h" contiene svariati #define e la defi-

nizione della struttura StructData. I file possono essere compilati utilizzando indifferentemente il compilatore della Manx o quello della Lattice.

Nel digitare i listati riportati di seguito, bisogna prestare particolare attenzione all'array 'structdata' che troviamo in ogni funzione di stampa delle strutture. Il primo carattere del nome di ogni membro è costituito da uno spazio, oppure da un meno, oppure da una parentesi tonda aperta. È importante usare quello giusto. Inoltre, è critico anche l'uso della costante che indica le dimensioni dei dati trattati, in quanto se viene usata quella sbagliata (esempio, INTSIZE al posto di PTRSIZE) otterremo dei risultati sbagliati, se non addirittura una visita del guru.

Note al programma

SB è già utile così com'è, ma alcuni vorranno modificarlo per aggiungere strutture alle quali sono particolarmente interessati. Ecco quindi, per facilitare questa operazione, una spiegazione sul funzionamento del programma.

Il principio che governa SB è molto semplice. Esiste una funzione specializzata per ogni tipo di struttura contemplata. Se una struttura contiene più membri di quanti ne siano visualizzabili sullo schermo contemporaneamente, ci sarà un'altra funzione per ciascuna delle pagine successive. Alla funzione viene passato un puntatore alla struttura in esame (e un offset, nel caso delle funzioni che riguardano le pagine successive) e questa ne stampa i nomi dei membri, i tipi e i valori, quindi attende un input. A seconda del membro selezionato dall'utente, un'apposita routine viene chiamata per stampare quel membro della struttura e quindi si attende nuovamente un input. Potrebbe darsi il caso in cui una routine chiama sé stessa, come succede nel passare attraverso una catena di strutture Gadget, per esempio.

In ogni caso, qualsiasi selezione effettuata porta il programma a un livello più in basso e l'unico modo che l'utente ha di ritornare verso l'alto è quello di selezionare il gadget "torna indietro" o "pagina preced", causando l'uscita dall'ultima funzione chiamata (dal momento che ogni livello che visitiamo rappresenta un livello in più nella nidificazione delle chiamate alle funzioni, c'è la possibilità che lo stack vada in overflow se si scende tropppo in basso. Abbiamo calcolato, tuttavia, che anche con il limitato stack di sistema di 4000 byte, bisognerebbe andare oltre 100 livelli più in basso per correre questo rischio).

Oltre alle funzioni specifiche per ogni struttura, ci sono le funzioni per gestire l'output esadecimale e la stampa esplicita dei flag. Queste sono contenute nel file "sb.c". In questo file troviamo anche la funzione put(), utilizzata per comunicare alla funzione di output in "sbio.c" che tipo di dati stampare. Questa viene chiamata da tutte le funzioni di output e le viene passato un puntatore alla struttura e un puntatore a un array di strutture StructData. La modifica di questo array è la chiave che permette di implementare nuove strutture in SB.

La struttura StructData, definita in "sb.h", contiene tutte le informazioni a proposito del membro di una struttura: il nome, il tipo, le modalità di stampa e le dimensioni. Il nome e il tipo sono semplicemente dei puntatori a stringhe da stampare. Le modalità di stampa sono necessarie per sapere come stampare i valori di quel membro: come byte, short o long, con o senza segno, o come una stringa o come un puntatore. Riferitevi alla funzione put() per sapere esattamente come viene interpretato questo argomento. L'ultimo membro della struttura StructData, la dimensione, indica il numero di byte utilizzati dal membro sotto esame. Le costanti BYTESIZE, INTSIZE e PTRSIZE, definite in "sb.h", specificano le dimensioni di 1, 2, e 4 byte. Quando il membro in esame è una struttura (e non un puntatore a essa), si utilizza la macro SZ, definita in "sb.h", per calcolarne le dimensioni. Ad esempio, SZ(View) anziché sizeof(struct View).

Prendendo spunto dalle funzioni PrWindow(), PrScreen() e PrGadget() non dovrebbe essere difficile aggiungere le altre. Quando implementiamo una nuova struttura, dobbiamo togliere le parentesi dal suo nome per permettere all'utente di selezionarla. Una parentesi tonda aperta come primo carattere del nome di un membro fa in modo che l'utente non possa selezionarlo (ne impedisce la trasformazione in gadget). Per renderne possibile l'esplorazione, basta togliere queste parentesi da tutte le strutture che contengono un puntatore alla nuova struttura appena implementata e aggiungere una chiamata alla nostra nuova funzione nel corpo dell'istruzione SWITCH.

In ogni funzione di stampa che implementiamo, dovremo preparare un array statico di strutture StructData, così come avviene nelle altre funzioni. I nomi dei membri devono iniziare con uno spazio nel caso di flag, array o puntatori a strutture che SB è in grado di gestire, con un meno (-) per semplici elementi di dati che devono essere visualizzati in nero e che non saranno selezionabili, oppure con una parentesi per i puntatori a strutture non ancora implementate.

La stampa del tutto e l'attesa di input dall'utente vengono gestite dalla funzione GetChoice(). I membri che l'utente può selezionare sono implementati come gadget privi di qualsiasi immagine o bordo. Un array di 16 strutture IntuiText è predisposto per ricevere il testo di ogni linea che deve essere stampata nella finestra. Viene preparato anche un array di 16 gadget di tipo booleano, ognuno dei quali punta a una diversa struttura IntuiText. Ci sono anche le strutture relative ai gadget "torna indietro" (che viene anche usato per "pagina preced") e "ancora".

Quando GetChoice() viene chiamata, essa chiama a sua volta Redisplay() per preparare i nomi dei membri della struttura e per costruire la necessaria lista di gadget. Redisplay() rimuove per prima cosa tutti i gadget esistenti eccetto il primo nella lista, il gadget "torna indietro". Esegue poi un loop tante volte quante sono le linee da stampare (massimo 16) per aggiornare il contenuto di IntuiText o per aggiungere un gadget alla lista. Dopo il loop viene aggiunto il gadget "ancora" (qualora ce ne sia bisogno). Infine viene chiamata la funzione di Intuition RefreshGadgets() per visualizzare il tutto nella finestra di SB.

Dopo che Redisplay() ha finito il suo lavoro, GetChoice() attende un evento IDCMP che segnala l'avvenuta selezione di un gadget da parte dell'utente. Se la finestra è stata chiusa, verrà

chiamata la funzione CloseOut() per ripulire tutto e chiudere le librerie prima di uscire. Se è stato selezionato un altro gadget, invece, la sua identità viene restituita alla funzione che ha chiamato GetChoice(). I numeri di identità dei gadget associati ai membri di una struttura sono costituiti dal loro numero sequenziale - 1 per il primo e così via. Il gadget "pagina preced" ha ID = 0, mentre "ancora" ha come ID la costante MOREGAD, definita come 25 nel file "sb.h".

Divertitevi con Structure Browser!

Listato 1:

```
/* sb.h */

#include <intuition/intuition.h>
#include <intuition/intuitionbase.h>
#include <stdio.h>

#define SZ(x)      sizeof(struct x)
#define DATASIZE   (sizeof(structdata) /
sizeof(struct StructData))
#define PTRSIZE    sizeof(APTR)
#define INTSIZE    sizeof(short)
#define BYTESIZE   sizeof(char)
#define MAXGADG    16
#define MOREGADG  25 /* ID del gadget "ancora"*/

struct StructData {
   char *membername;
   char *membertype;
   int  printtype;
   int  datasize;
   };
```

Listato 2:

```
/* The Transactor Structure Browser (SB) V1.0
 *
 * By Nick Sullivan and Chris Zamara (AHA!) (c)1989
 *
 * Strutture implementate nella V1.0:
 * IntuitionBase, Window, Screen, RastPort, BitMap,
 * Gadget, Node, MsgPort, IntuiText, TextAttr.
 *
 * Il programma può essere distribuito liberamente.
 */

#include "sb.h"
#define MIN(x,y) ((x)<(y)?(x):(y))
#define FLAGFIELDS 4
extern struct IntuitionBase *IntuitionBase;
extern struct IntuiText ChoiceText[], BackIText;
APTR OpenLibrary ();
int level = 0; /* current level of nesting */
static char textlines[MAXGADG + 1][80];
extern void PrIntuiBase(), HexLine();

void main()
{
int choice = -1;
   SetupGadg();
   OpenStuff(); /* apre intuition.library & window */
   while (choice) {
     putHeader("Scegliere una libreria", NULL);
     ChoiceText[0].IText = (UBYTE *)" Intuition
                  struct Library";
     BackIText.IText = (UBYTE *)" fine programma";
     switch (choice = GetChoice(1)) {
       case 1:
         PrIntuiBase ("struttura IntuitionBase",
                  IntuitionBase);
         break;
       }
   }
   CloseOut();
}

void PrIntuiBase (string, IBase) char *string;
struct IntuitionBase *IBase;
{
   static struct StructData structdata[] = {
     { "(LibMode",          "struct Library)", 0,
                   SZ(Library) },
     { "(ViewLord",          "struct View)",   0,
                   SZ(View)      },
     { " ActiveWindow",      "struct Window *", 5,
                   PTRSIZE       },
     { " ActiveScreen",      "struct Screen *", 5,
                   PTRSIZE       },
     { " FirstScreen",       "struct Screen *", 5,
                   PTRSIZE       }
   };
int sum, choice = -1;
   level++;
   while (choice) {
     sum = SetOptionText(string, structdata,
                  (APTR)IBase, DATASIZE, 0);
     switch (choice = GetChoice(DATASIZE)) {
       case 3:
         if (IBase->ActiveWindow)
           PrWindow("Window attualmente attiva",
                  IBase->ActiveWindow);
         break;
       case 4:
         PrScreen("Screen attualmente attivo",
                  IBase->ActiveScreen);
         break;
       case 5:
```

I dischi di Fred Fish

Prima guida ragionata al loro contenuto

ANIMAZIONE

53	3D_PYRAMID			CREA PIRAMIDI COLORATE E LE RUOTA IN 3D - (PLAYER)	
112	BEACHBIRDS			SCENA RAFFIGURANTE UNA SPIAGGIA CON GABBIANI CHE VOLANO	[TUNNELL]
100	BERSERK			MONOCICLI CHE RIMBALZANO SU UNA SFERA - (SHOWANIM)	[SCHWAB]
132	BERSERK		*	MONOCICLI CHE RIMBALZANO SU UNA SFERA - (SHOWANIM)	[SCHWAB]
134	BOINGTHROWS			CIOCATORE CHE FA CANESTRO	[LANDIS]
53	CAMERA			ANIMAZIONE DI UN CAMMELLO CHE CAMMINA	[AEGIS DEVELOPMENT]
123	CAR			INCIDENTE AUTOMOBILISTICO - (SHOWANIM)	[HASTINGS]
53	CLOWN			CLOWN CHE CHE IMITA UN GIOCOLIERE CONTRE BIRILLI	[AEGIS DEVELOPMENT]
106	DEATHSTAR			ASTRONAVI E MORTE NERA / (DA GUERRE STELLARI)	
125	ELGATO	1.4		GATTO CHE CAMMINA / (RUOTA IN 3D) - (SHOWANIM)	[BLAIR-SULLIVAN]
116	F-15			ANIMAZIONE DI UN F-15 DISEGNATO PER LINEE - (MOVIE)	[GRAHAM]
112	FOCUS			FUNZIUONI MATEMATICHE SI DISSOLVONO ...	[HAGEN]
29	GROW-UR-OWN			ROBOT CREA UN ROBOT FEMMINA - (PLAYER)	[AEGIS DEVELOPMENT]
126	HBHILL			UOMO CHE BALLA / (MODALITA' HALF BRITE) - (SHOWANIM)	[BLAIR-SULLIVAN]
47	JUGGLER	1.0		GIOCOLIERE CON TRE SFERE / (RAY-TRACED)	[GRAHAM]
97	JUGGLER	1.0		GIOCOLIERE CON TRE SFERE / (RAY-TRACED)	[GRAHAM]
116	KAHNANKAS			SIMULAZIONE DI MOTO PERPETUO / (RAY-TRACED)	[OFFER]
115	KILLER			AMIGA CONTRO LE COMPETIZIONI	[WILT]
29	KNIGHT			CAVALIERE CHE CAVALCA UN CAVALLO - (PLAYER)	[SACHS]
106	LASERS			LASER CHE DISEGNA LA SCRITTA QUICKFLIX - (QUICKFLIX)	
73	LMV			JIMMY STEWART / (SEQUENZE TV DIGITALIZZATE) - (LMV)	[WEBSTER]
29	LUNA			ATTERRAGGIO SULLA LUNA - (PLAYER)	[AEGIS DEVELOPMENT]
109	MACHINE			MACCHINA ANIMATA - (SHOWANIM)	[HASTINGS]
115	MARKETROID		*	COMMERCIALIZZAZIONE DI AMIGA / (IN FORMA DI DEMO GIOCO)	[SCHWAB]
127	NEMESIS			STELLE PIANETI E BUCHI NERI / (CON EFFETTI SONORI)	[RILEY]
126	ONLYAMIGA			PIRAMIDE COMPOSTA DA TRE SFERE	[HANS SINGH]
112	RGB			MOSTRA I RISCHI DELLE RADIAZIONI RGB - (PROJECTOR)	[HAGEN]
127	RIPPLES			BANDIERA OSSERVATA DA VARI PUNTI DI VISTA - (SHOWANIM)	[HANSTINGS]
116	ROCKER			ANIMAZIONE CON OMBRE DI UNA SEDIA - (MOVIE)	[GRAHAM]
29	RUBIK		*	CUBO DI RUBIK ANIMATO IN 3D	[KINNERLEY]
29	SNAPDEMO			MONTA E SMONTA UN PUZZLE - (PLAYER)	[AEGIS]
106	SWING			SIMULAZIONE DI MOTO PERPETUO - (QUICKFILX)	[QUICKFLIX]

COMANDI BATCH

39	ASK		*	RICHIEDE UNA RISPOSTA DI UN SINGOLO CARATTERE	[WILLIAMS]
46	CHECKMODEM			CONTROLLA LA PRESENZA DI UN MODEM	[DILLON]
123	ELSE			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	ENDIF			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
29	ENOUGH		*	TEST PER LA PRESENZE DELLE RISORSE ADEGUATE / (I.E. RAM)	[BARRETT]
123	FAILAT			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	IF			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
32	INPUT	1.1		SCRIVE L'INPUT DI UN UTENTE IN UN FILE DELLA RAM	[BEOGELEIN]
123	LAB			ARP - SOSTITUSCE IL COMANDO AMIGADOS	
79	MOUNTED		*	CONTROLLA SE IL DEVICE SPECIFICATO E' STATO MONTATO	[NESBITT]
79	MOUNTED		*	CONTROLLA SE IL DEVICE SPECIFICATO E' STATO MONTATO	[DE SILVA]
49	QMOUSE		*	RITORNA IL VALORE DEL PULSANTE SINISTRO DEL MOUSE	[RETHMEYER]

79	QUERYANY	2.0	*	PERMETTE DI GESTIRE DECISIONI DEL TIPO SI/NO	[SMITH]
123	QUIT			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	SKIP			ARP - SOSTITUISCE IL COMANDO AMIGADOS	

COMANDI CLI

123	ADDBUFFERS			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
39	ALIST		*	SOSTITUISCE IL COMANDO AMIGADOS / (SOLO NOME DEI FILE)	[WILLIAMS]
32	AREACODE	1.4		DECODIFICA I PREFISSI DEGLI USA	[BEOGELEIN]
40	AREACODE	1.5		DECODIFICA I PREFISSI DEGLI USA	[BEOGELEIN]
32	ASC	1.1		RITORNA IL VALORE ASCII DI UN CARATTERE	[BEOGELEIN]
52	ASSIGN		*	SOSTITUISCE IL COMANDO AMIGADOS	[MCMANIS]
32	BIT	1.1		ESEGUE OPERAZIONI LOGICHE COME AND,OR,XOR,EQU & IMP	[BEOGELEIN]
123	BREAK			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	BREAK			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	BREAK	87/01	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[BALLANTYNE]
123	CD			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	CD	87/01	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[HEATH]
123	CHANGETASKPRI			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	CHMOD	87/01	*	ARP - CAMBIA LO STATO DI PROTEZIONE DI UN FILE	[BALLANTYNE]
32	CHR	1.1		RITORNA IL VALORE ESAD.,BIN.,DEC.,OTT., DI UN VALORE	[BEOGELEIN]
39	CLS		*	CANCELLA IL CONTENUTO DELLA FINESTRA CORRENTE	[WILLIAMS]
32	CONV	1.1		CONVERSIONI DECIMALI,ESADECIMALI,OTTALI,BINARIE	[BEOGELEIN]
75	COPY	1.0	*	SOSTITUISCE IL COMANDO AMIGADOS / (NON CAMBIA LE DATE)	[LYDIATT]
123	CP			ARP - SOSTITUISCE IL COMANDO COPY DI AMIGADOS	
123	DELETE			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
36	ECHO			SOSTITUISCE IL COMANDO ECHO	[PHILLIPS]
39	ECHO		*	SOSTITUISCE IL COMANDO ECHO / (PERMETTE \N)	[WILLIAMS]
79	ECHO		*	SOSTITUISCE IL COMANDO ECHO / (PERMETTE ESA.,BIN.,DEC.,)	[NESBITT]
123	ECHO			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	ECHO	86/12	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[HEATH]
123	FILENOTE			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	FILENOTE	87/01	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[BALLANTYNE]
32	HELP	1.1		RITORNA LA SINTASSI DEI COMANDI AMIGADOS/DOSPLUS	[BEOGELEIN]
79	INFO		*	SOSTITUISCE IL COMANDO AMIGADOS	[MCMANIS]
123	INFO			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
87	INSTALL		*	SOSTITUISCE IL COMANDO AMIGADOS	[TURNER]
87	INSTALL		*	SOSTITUISCE IL COMANDO AMIGADOS	[NESBITT]
123	JOIN			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
105	L			SOSTITUISCE IL COMANDO AMIGADOS / (MOSTRA FLAG ARCHIVIO)	[DAVIES]
123	MAKEDIR			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	MAKEDIR	87/01	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[HEATH]
32	MATH	1.2		ESEGUE SOMMME,SOTTRAZIONI,DIVISIONI,MOLTIPLICAZIONI	[BEOGELEIN]
38	NEWSTAT		*	SOSTITUISCE IL COMANDO AMIGADOS	[SYNGE]
123	PATH			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	PROMPT			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
53	PROMPT	87/01	*	ARP - SOSTITUISCE IL COMANDO AMIGADOS	[HEATH]
123	PROTECT			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	RELABEL			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	RENAME			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
6	SETPARALLEL	1.1	*	CAMBIA I PARAMETRI DELLA PORTA PARALLELA DA CLI	[POHORSKY (STOBIE)]
6	SETSERIAL	3.2	*	CAMBIA I PARAMETRI DELLA PORTA SERIALE DA CLI	[POHORSKY (STOBIE)]
123	SORT			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
105	STACK		*	SOSTITUISCE IL COMANDO AMIGADOS / (IN ASSEMBLER)	[MCDIARMID]
123	STACK			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	STATUS			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
123	TYPE			ARP - SOSTITUISCE IL COMANDO AMIGADOS	
79	WHY	1	*	SOSTITUISCE IL COMANDO AMIGADOS / (PIU' INFORMAZIONI)	[NESBITT]
74	XCOPY			SOSTITUISCE IL COMANDO AMIGADOS	[ROBERTSON]

COLLEZIONE COMANDI CLI

79	ASMTOOLS	*	5 COMANDI SOSTITUTIVI	[NESBITT]

COMANDI CLI

32	DOSPLUS1	COMANDI CLI UTILI AI SVILUPPATORI DI SOFTWARE	[BEOGELEIN]
32	DOSPLUS2	COMANDI CLI UTILI AI SVILUPPATORI DI SOFTWARE	[BEOGELEIN]

UTILITY CLI

Disco	Nome	Versione		Descrizione	Autore
74	CLED	1.3		EDITOR DI LINEA	[EMPLEO]
81	CLED	1.4		EDITOR DI LINEA	[EMPLEO]
69	CONMAN	0.9	*	CON:EDITOR DI LINEA, L'HISTORY ECC.	[HAWES]
90	CONMAN	0.98B		CON:EDITOR DI LINEA, HISTORY ECC.	[HAWES]
81	CONMAN	0.99B		CON:EDITOR DI LINEA, HISTORY ECC.	[HAWES]
100	CONMAN	1.0		CON:EDITOR DI LINEA, HISTORY ECC.	[HAWES]
133	CONMAN	1.1		CON:EDITOR DI LINEA, HISTORY ECC.	[HAWES]
40	DOSHELP	1.60	*	ASSISTANZA NELL'USO DEI COMANDI AMIGADOS	[YOUELLS]
50	FIRSTSILICON	8611.2		PERMETTE DI IMMETTERE I COMANDI AMIGADOS DA UNA FINESTRA	[GOODEVE]
75	HARDCOPY		*	PRODUCE L'HARDCOPY DEL CLI CORRENTE	[CERVONE]
131	MACKIE		*	PARTE UN CLI IN QUALSIASI MOMENTO	[ROKICKI]
71	PETCLI			PERMETTE DI EDITARE UNA LINEA CLI - (AMIGABASIC)	[KITTEL]
35	POPCLI	1.0	*	PARTE UN CLI IN QUALSIASI MOMENTO / (SALVA SCHERMO)	[TOEBES]
40	POPCLI	2.0	*	PARTE UN CLI IN QUALSIASI MOMENTO / (SALVA SCHERMO)	[TOEBES]
84	POPCLI	III	*	PARTE UN CLI IN QUALSIASI MOMENTO / (SALVA SCHERMO)	[TOEBES]
65	RUNBACK		*	APRE UN CLI E PARTE IL PROGRAMMA ALLA FINE CHIUDE IL CLI	[HEATH]
73	RUNBACKGROUND		*	APRE UN CLI E PARTE IL PROGRAMMA ALLA FINE CHIUDE IL CLI	[PECK]
102	SILICON			PICCOLA LINEA DI INPUT PER GESTIRE I COMANDI AMIGADOS	[GOODEVE]
43	WBRUN		*	PERMETTE DI FAR PARTIRE I PROGRAMMI WBENCH DA CLI	[TOEBES]

COPIATORI DI DISCHI

Disco	Nome	Versione		Descrizione	Autore
98	BACKUP		*	BACKUP DA HARD DISK A FLOPPY / (MOLTO SEMPLICE)	[KENT]
82	D2D-DEMO	2.0		DEMO DI DISK-2-DISK / (DA CBM DOS2.1 A AMIGA)	[CENTRAL COAST]
131	DFC	1	*	FORMATTA/COPIA	[ROKICKI]
109	DOSKWIK	2.0		COPIATORE VELOCE DI RAM A DISCO E VICEVERSA	[KEMPER]
129	DOSKWIK	2.0		COPIATORE VELOCE DI RAM A DISCO E VICEVERSA	[KEMPER]
128	MRBACKUP	1.1	*	BACKUP DA HARD DISK A FLOPPY / (COMPRIME I DATI)	[RINFRET]
129	MRBACKUP	2.0	*	BACKUP DA HARD DISK A FLOPPY / (COMPRIME I DATI)	[RINFRET]
129	MRBACKUP	2.1		BACKUP DA HARD DISK A FLOPPY / (COMPRIME I DATI)	[RINFRET]
49	MYUPDATE		*	COPIA FILE CHE ESISTONO GIA' NELLA DESTINAZIONE	[MUELLER]
35	QUICKCOPY	1.0A/EA		COPIATORI PER DISCHI	[DEVENPORT]
128	SDBACKUP	1.1		BACKUP DA HARD DISK A FLOPPY / (USA I BIT DI ARCHIVIO)	[DREW]
45	UPDATE	1.7		COPIA I FILE PIU' NUOVI DEI FILE DI DESTINAZIONE	

EDITOR DI DISCHI

Disco	Nome	Versione		Descrizione	Autore
50	DISKZAP	1.1		EDITOR DI DISCHI	[BINGHAM JR]
58	NEWZAP	3.0	*	EDITOR DI DISCHI	[HODGSON]
102	SECTORAMA	1.0		EDITOR DI DISCHI / (VISUALIZZA UN SETTORE INTERO)	[JOINER]
108	SECTORAMA	1.1		EDITOR DI DISCHI / (VISUALIZZA UN INTERO SETTORE)	[JOINER]

ASSEMBLER 68000/10

Disco	Nome	Versione		Descrizione	Autore
110	A68K	1.02	*	ASSEMBLER PER 68000	[GIBBS]
46	ASM	1.0		MACRO ASSEMBLER PER 68010	[LEAVITT]
50	ASM	1.1		MACRO ASSEMBLER PER 68010	[LEAVITT]
66	ASM68K	1.0.1		MACRO ASSEMBLER PER 68000	[HOWE]
69	ASM68K	1.0.3		MACRO ASSEMBLER PER 68000	[HOWE]
81	ASM68K	1.1.0		MACRO ASSEMBLER PER 68000	[HOWE]

UTILITY ASSEMBLER 68000

Disco	Nome	Versione		Descrizione	Autore
128	DIS		*	DISASSEMBLER PER 68000	[LEE]

LINGUAGGIO C

Disco	Nome	Versione		Descrizione	Autore
53	COMPILER		*	COMPILATORE C OTTIMIZZATO / (NON PIENAMENTE PORTABILE)	[BRANDT]
110	PDC	0.2	*	COMPILATORE C OTTIMIZZATO	[LYDIATT (BRANDT)]

LINGUAGGIO CP/M

109 SIMCPM 1.00 * EMULATORE CP/M 2.2 / (8080 CPU & TERMINALE H19) [CATHEY/GIBBS]

LINGUAGGIO DRACO

76 DRACO DISK 1 LINGUAGGIO DRACO / (DISCO DI SISTEMA) [GRAY]
77 DRACO DISK 2 LINGUAGGIO DRACO / (DISCO DATI) [GRAY]

LINGUAGGIO FORTH

3 CFORTH * IMPLEMENTAZIONE FACILMENTE PORTABILE DI FIGFORTH [PRATT]
9 MVP-FORTH 1.03A MOUNTAIN VIEW PRESS FORTH [FANTASIA SYSTEMS]

LINGUAGGIO LISP

3 XLISP 1.4 * PICCOLO INTERPRETE SPERIMENTALE DI LISP / (NON FUNZIONA) [BETZ]
18 XLISP 1.6 * PICCOLO INTERPRETE SPERIMENTALE DI LISP [BETZ]
39 XLISP 1.7 PICCOLO INTERPRETE SPERIMENTALE DI LISP [BETZ]

LINGUAGGIO LOGO

70 LOGO INTERPRETE LOGO / (SIMILE COME USO AL LOGO DELL'APPLEII) [QWENS]

LINGUAGGIO MODULA2

113 M2AMIGA VERSINE DEMO DI UN COMPILATORE IN UN SOLO PASSAGGIO [AMSOFT]
24 MODULA-2 1.1 VERSIONE INIZIALE DEL COMPILATORE A SINGOLO PASSO ETHZ

UTILITY PER 68000

136 SMALLTALK * SERIE DI MACRO PER MIGLIORARE L'INTERFACCIA CON AMIGADOS [LEE]

PROGRAMMAZIONE

138 AMIGALINE NOTE TECNICHE PER IL PROGRAMMATORE
66 ASSIGNED * PER CONTROLLARE SE UN NOME E STATO ASSEGNATO [GREEN]
103 AVLTREES 1.0 * ROUTINE PER CREARE E USARE ALBERI BINARI BILANCIATI [VIXIE]
34 BTREE 1.1 * ROUTINE PER IMPLEMENTARE ALBERI BINARI [HELLIER-JEFFERSON]
34 BTREE2 1.0 * LIBRERIA DI ROUTINE ORIGINALE PER ALBERI BINARI [HELLIER]
54 HANOI * TORRE DI HANOI / (DEMO SULLA PROGRAMMAZIONE RICURSIVA) [OZER]
1 HELLO * 'HELLO WORLD' FINESTRA DEMO DA RKM
5 INPUT.DEV * GESTISCE LE ECCEZIONI MOUSE/KEYBOARD PRIMA DI INTUITION [PECK]
5 JOYSTICK * RKM ESEMPIO SULL'UTILIZZO DELLA GAMEPORT CON IL JOYSTICK [PECK]
5 KEYBOARD * RKM ESEMPIO DI COMUNICAZIONE DIRETTA CON LA TASTIERA [PECK]
79 KILL 1.01 * ELIMINA TASK,CHIUDE FINESTRE, SCARICA CODICI [MUSSER]
111 LABELS AIUTO PER LE COSTANTI DI SISTEMA [SEIBERT]
61 LPATCH * MODIFICA ATOM & PROGRAM. CON MALFUNZIONANTE 1.0 LSTARTUP [SCHEPPNER]
18 MC68010 * SPIEGAZIONE ISTRUZIONI DEL 68010 E DIFFERENZE CON 68000 [FLORYAN-TURNER]
5 MOUSE * SETTA IL MOUSE NELLA GAMEPORT 2 [PECK]
55 NEWSTARTUPS * NUOVO SET DI MODULI STARTUP PER C [SCHEPPNER]
5 ONE.WINDOW * FINESTRA CON CONSOLE ATTACCATA [PECK]
87 PALTEST * ESEMPIO DI COME TESTARE SE E' UNA MACCHINA PAL [BONNKIRCH]
5 PARALLEL * ESEMPIO DI ACCESSO ALLA PORTA PARALLELA [POHORSKY]
73 PAROUT * ACCESSO ALLA PORTA PARALLELA TRAMITE CIAA./MISC.RESOURCE [LINDSAY]
92 PARSE * GESTISCE ESPRESSIONI / (COMPUTA & STAMPA I RISULTATI) [OLSEN]
20 PORTHANDLER * SEMPLICE PORT-HANDLER [TOEBES]
107 PROSUITE 1.01 * PER PROGRAMMATORI SUITE 1.01 [MICAL]
20 RANDOM * GENERATORE DI NUMERI CASUALI IN C [BEATS]
74 RANDOM * GENERATORE DI NUMERI CASUALI IN C [TOOLE]
85 RAWIO * ESEMPIO DI INPUT DA RAW & MODALITA CBREAK [MCMAINS]
5 TEXT.DEMO * RKM ESEMPIO MOSTRANDO VARI CARATTERI & ATTRIBUTI [PECK]

```
      PrScreen("Primo screen nella lista di
                 Intuition", IBase->FirstScreen);
      break;
    }
  }
  level--;
}


void put (option, stuff, base, offset)
int option; struct StructData *stuff; char *base;
int offset;
{
register long lnum;
register int  inum;
char buf[40];
int i;
  sprintf(textlines[option], "%-16s%-24s",
            stuff->membername, stuff->membertype);
  switch (stuff->printtype) {
    case 0:       /* non stampare niente */
      buf[0] = '\0';
      break;
    case 1:       /* stampa una long */
      lnum = *(long *)(base + offset);
      sprintf(buf, "$%8lx  %01d", lnum, lnum);
      break;
    case 2:       /* stampa uno short */
      inum = *(short *)(base + offset);
      sprintf(buf, "$%8x  %10d", inum, inum);
      break;
    case 3:       /* stampa un byte */
      inum = *(base + offset);
      sprintf(buf, "$%8x  %10d", inum, inum);
      break;
    case 4:       /* stampa una stringa */
      if (!(lnum = *(long *)(base + offset) ))
        sprintf(buf, "NULL");
      else {
        for (i = 0; i < 30 && *((char *)lnum + i);
                                        i++)
          buf[i + 1] = *((char *)lnum + i);
        buf[0] = buf[i + 1] = '\"';
        buf[i + 2] = '\0';
        if (*((char *)lnum + i))
          strcat(buf, "...");
      }
      break;
    case 5:       /* stampa un puntatore */
      if (!(lnum = *(long *)(base + offset) ))
        sprintf(buf, "NULL");
      else
        sprintf(buf, "$%8lx  %10ld", lnum, lnum);
      break;
    case 11:      /* stampa una long */
      lnum = *(long *)(base + offset);
      sprintf(buf, "$%8lx  %10lu", lnum, lnum);
      break;
    case 12:          /* stampa uno short */
      inum = *(short *)(base + offset);
        sprintf(buf, "$%8x  %10u", inum, inum);
        break;
      case 13:        /* stampa un byte */
        inum = *(base + offset);
        sprintf(buf, "$%8x  %10u", inum, inum);
        break;
    }
    strcat(textlines[option], buf);
    ChoiceText[option].IText = (UBYTE *)
                   textlines[option];
}


void FlagPrint(string, names, flags)
char *string, **names; ULONG flags;
{
int i, line, fields = FLAGFIELDS;
char buf[32];
  SetBackText(1);  /* 'torna indietro' */
  for (i = 0; i < 8; i++) {
    strcpy(textlines[i], "-");
    ChoiceText[i].IText = (UBYTE *)textlines[i];
  }
  putHeader(string, NULL);
  for (i = line = 0; i < 32; i++) {
    if ((flags & (1L << i)) && names[i]) {
      sprintf(buf, "%-19s", names[i]);
      strcat(textlines[line], buf);
      if (!--fields) {
        ChoiceText[line].IText = (UBYTE *)
                  textlines[line];
        line++;
        fields = FLAGFIELDS;
      }
    }
  }
  if (fields < FLAGFIELDS)
    ChoiceText[line].IText = (UBYTE *)
                  textlines[line];
  while (GetChoice(line + 1))
    ;
}


void HexDump(string, address, unit, size)
char *address, *string; int unit; long size;
{
int line = 0, c;
char *buf[80];
  BackIText.IText = (UBYTE *)" interrompi hex ";
  if (size == -1)
    size = 0x7ffff;
  do {
    sprintf(buf, "%s da %lx (%ld)", string,
                  address, address);
    putHeader(buf, NULL);
    if (line == MAXGADG)
      line = 0;
    while (line < MAXGADG && size > 0) {
      HexLine(address, unit, line++, size);
```

```
      size -= 16;
      address += 16;
    }
    c = GetChoice(size > 0 ? MAXGADG + 1 : line);
  } while (size > 0 && c == MOREGADG);
}


void HexLine (address, unit, line, size)
UBYTE *address; int unit, line; long size;
{
USHORT i, j;
char buf[80];
static char hexdigit[] = "0123456789ABCDEF";
  sprintf(textlines[line], "-%6lx: ", address);
  for (i = 0; i < MIN(size, 16); i += unit) {
    switch (unit) {
      case BYTESIZE:
        j = *(address + i);
        sprintf(buf, "%c%c ", hexdigit[j / 16],
                 hexdigit[j % 16]);
        break;
      case INTSIZE:
        sprintf(buf, "%04x ", *(short *)(address +
                 i));
        break;
      case PTRSIZE:
        sprintf(buf, "%08lx ", *(long *)(address +
                 i));
        break;
    }
    strcat(textlines[line], buf);
  }
  ChoiceText[line].IText = (UBYTE *)textlines[line];
}


int SetOptionText (hdrtext, data, object, size,
offset)
char *hdrtext;
struct StructData *data;
APTR object;
int size, offset;
{
int i, sum;
  SetBackText( offset ? 1 : 0);
  putHeader(hdrtext, object);
  for (i = sum = 0; i < size; i++) {
    put(i, &data[i], object, sum + offset);
    sum += data[i].datasize;
  }
  return (sum + offset);
}


void PrString(heading, string) char *heading,
*string;
{
char *newstring, *malloc();
  putHeader(heading, NULL);
  newstring = malloc(strlen(string) + 1);
  *newstring = '-';
  strcpy(newstring + 1, string);
  ChoiceText[0].IText = (UBYTE *)newstring;
  GetChoice(1);
  free(newstring);
}
```

Listato 3:

```
#include "sb.h"

#define     CHOICEWIDTH 280
#define     CHOICEHEIGHT 8
#define     SPACING 9
#define     TOPGADG 30
#define     PUTTEXT(text, x, y) { Move(rp, (long)x,
(long)y); \
                              Text(rp, text,
(long)strlen(text)); }
struct   IntuitionBase  *IntuitionBase = NULL;
struct   GfxBase        *GfxBase = NULL;
struct   Window         *OpenWindow(), *MainWindow =
                  NULL;
struct   RastPort       *rp;
extern   int            level;
APTR  OpenLibrary();
struct   IntuiText   ChoiceText[MAXGADG + 1];
struct   Gadget      ChoiceGadg[MAXGADG + 1];
extern   void        CloseOut(), Redisplay();

struct IntuiText BackIText = {
        3, 2, JAM2,
        5, 2,
        NULL, NULL, NULL
    };
struct IntuiText MoreIText = {
        2, 3, JAM2,
        5, 2,
        NULL,
        (UBYTE *)"ancora",
        NULL
    };
struct Gadget BackGadg = {
        NULL,
        10, -12, 140, 12,
        GADGHCOMP | GRELBOTTOM,
        RELVERIFY,
        BOOLGADGET,
        NULL, NULL, &BackIText,
        NULL, NULL,
        0, NULL /* gadget ID è zero */
    };
struct Gadget MoreGadg = {
        NULL,
        300, -12, 59, 12,
        GADGHCOMP | GRELBOTTOM,
```

```
        RELVERIFY,
        BOOLGADGET,
        NULL, NULL, &MoreIText,
        NULL, NULL,
        MOREGADG, NULL
    };
struct NewWindow NWindow = {
        0, 10, 640, 189,        /* left, top,
                  width, height */
        -1, -1,                 /* usa i colori
                  dello screen */
        GADGETUP        /* IDCMP flags */
         | CLOSEWINDOW,
        WINDOWDEPTH     /* window flags */
         | WINDOWCLOSE
         | WINDOWDRAG
         | RMBTRAP
         | ACTIVATE
         | NOCAREREFRESH
         | SMART_REFRESH,
        &BackGadg,      /* primo gadget nella lista */
        NULL,
        (UBYTE *)"The Transactor Structure Browser
                  V 1.0",
        NULL, NULL,
        0, 0, 0, 0,
        WBENCHSCREEN
    };


void SetupGadg()
{
int i;
  for (i = 0; i < MAXGADG; i++) {
    ChoiceText[i].BackPen      = 0;
    ChoiceText[i].DrawMode     = JAM2;
    ChoiceText[i].LeftEdge     = 0;
    ChoiceText[i].TopEdge      = 0;
    ChoiceText[i].ITextFont    = NULL;
    ChoiceText[i].IText        = NULL;
    ChoiceText[i].NextText     = NULL;
    ChoiceGadg[i].LeftEdge     = 20;
    ChoiceGadg[i].TopEdge      = i * SPACING +
                    TOPGADG;
    ChoiceGadg[i].Width        = CHOICEWIDTH;
    ChoiceGadg[i].Height       = CHOICEHEIGHT;
    ChoiceGadg[i].Flags        = GADGHCOMP;
    ChoiceGadg[i].Activation   = RELVERIFY;
    ChoiceGadg[i].GadgetType   = BOOLGADGET;
    ChoiceGadg[i].GadgetText   = &ChoiceText[i];
    ChoiceGadg[i].GadgetID     = i + 1;  /* gadget
                    ID incomincia da 1 */
  }
}


int GetChoice (num) int num;
{
struct IntuiMessage *GetMsg(), *message;
ULONG msgclass;  /* classe del messaggio di IDCMP */
APTR IAddr;       /* puntatore al gadget fornito da
IDCMP */
  Redisplay(num);  /* prepara le scelte nella
                      finestra */
  FOREVER {     /*** loop principale ***/
    Wait (1L << MainWindow->UserPort->mp_SigBit);
    while (message = GetMsg(MainWindow->UserPort)) {
                  /* prendi dalla message port
                      quello che ci serve */
      msgclass = message->Class;
      IAddr    = message->IAddress;
      ReplyMsg(message);   /* rispondi subito al
                  messaggio */
            /* verifica se è stato selezionato un
                  gadget */
      if (msgclass == GADGETUP)
        return ( (int)((struct Gadget *)
                  IAddr)->GadgetID );
            /* chiudi tutto se è stato selezionato
                  CLOSEWINDOW */
      else if (msgclass == CLOSEWINDOW)
        CloseOut(); /* pulisci ed esci */
    }
  }
  return(0);         /* dummy */
}


void putHeader(string, ptr) char *string; APTR ptr;
/* metti titolo e puntatore in cima allo screen -
 * se ptr == NULL, metti solo la stringa. */
{
char buf[80];
  SetAPen(rp, 0L);
  RectFill(rp, 1L, 10L, (long)MainWindow->Width-25,
                  27L);
  SetAPen(rp, 3L);
  if (ptr) {
    sprintf(buf, "%d: %s (address $%lx):", level,
                  string, ptr);
    PUTTEXT(
     " Membro         Tipo
                  Valore(hex/decimal)",
     20L, 10L + 2 * rp->TxHeight);
  }
  else
    sprintf(buf, "%d: %s:", level, string);
  PUTTEXT(buf, 20L, 10L + rp->TxHeight);
}


void OpenStuff ()
{
  if (!(IntuitionBase = (struct IntuitionBase *)
           OpenLibrary("intuition.library", 0L) ) )
    CloseOut();
  if (!(GfxBase = (struct GfxBase *)
           OpenLibrary("graphics.library", 0L) ) )
    CloseOut();
  if (!(MainWindow = OpenWindow(&NWindow)) )
```

```
  CloseOut();
  rp = MainWindow->RPort; /* rastport per le
       routine grafiche */
}


void CloseOut()
{
  /* chiudi tutto prima di uscire */
  if (MainWindow)     CloseWindow(MainWindow);
  if (IntuitionBase) CloseLibrary(IntuitionBase);
  if (GfxBase)        CloseLibrary(GfxBase);
  exit(0);
}


void Redisplay(num) int num;
   /* pulisci la window, rimuovi i vecchi gadget,
   prepara e aggiungi quelli nuovi */
{
struct Gadget *gadg;
BOOL MoreFlag = FALSE;
int i, c;
  SetAPen(rp, 0L); /* rectfill con colore di
          background per pulire */
  RectFill(rp, 1L, (long)TOPGADG,
          (long)MainWindow->Width - 2,
          (long)MainWindow->Height - 2);
  if (num > MAXGADG) {
   num = MAXGADG;
   MoreFlag = TRUE; /* prepara il gadget "ancora" */
  }
  /* rimuovi tutti i gadget delle opzioni */
  gadg = &BackGadg;
  while (gadg = gadg->NextGadget)
    RemoveGadget(MainWindow, gadg);
  /* visuakizza i gadget a seconda del codice che
  * si trova all'inizio dell'IText di ogni gadget */
  for (i = 0; i < num; i++) {
   ChoiceText[i].FrontPen = 1;
   if ((c = *ChoiceText[i].IText) == '-' || c == '(')
      {
      if (c == '-') {
        *ChoiceText[i].IText = ' ';
        ChoiceText[i].FrontPen = 2;
      }
      PrintIText(rp, &ChoiceText[i],
           (long)ChoiceGadg[i].LeftEdge,
           (long)ChoiceGadg[i].TopEdge);
    }
    else
      AddGadget(MainWindow, &ChoiceGadg[i], -1L);
  }
  if (MoreFlag)
    AddGadget(MainWindow, &MoreGadg, -1L);
  /* visualizza i gadget (il testo) */
  RefreshGadgets(&BackGadg, MainWindow, NULL);
}
```

```
void SetBackText (sflag) int sflag;
{
  BackIText.IText = (UBYTE *)(sflag ?
          " pagina preced  " : " torna indietro ");
}
```

Listato 4:

```
#include "sb.h"
extern int level;


void PrGadget(string, gadget) char *string; struct
Gadget *gadget;
{
static struct StructData structdata[] = {
  { " NextGadget",  "struct Gadget *",  5, PTRSIZE },
  { "-LeftEdge",    "SHORT",            2, INTSIZE },
  { "-TopEdge",     "SHORT",            2, INTSIZE },
  { "-Width",       "SHORT",            2, INTSIZE },
  { "-Height",      "SHORT",            2, INTSIZE },
  { " Flags",       "USHORT",          12, INTSIZE },
  { " Activation",  "USHORT",          12, INTSIZE },
  { " GadgetType",  "USHORT",          12, INTSIZE },
  { "(GadgetRender", "APTR)",           5, PTRSIZE },
  { "(SelectRender", "APTR)",           5, PTRSIZE },
  { " GadgetText", "struct IntuiText *", 5, PTRSIZE },
  { "-MutualExclude","LONG",            1, PTRSIZE },
  { "(SpecialInfo", "APTR)",            5, PTRSIZE },
  { "-GadgetID",    "USHORT",          12, INTSIZE },
  { "-UserData",    "APTR",             5, PTRSIZE }
  };
static char *flagnames[16] = {
     "GADGHBOX",      "GADGHIMAGE",
"GADGIMAGE",      "GRELBOTTOM",
     "GRELRIGHT",     "GRELWIDTH",
"GRELHEIGHT",     "SELECTED",
     "GADGDISABLED"
  };
static char *activatenames[16] = {
     "RELVERIFY",     "GADGIMMEDIATE",
            "ENDGADGET",      "FOLLOWMOUSE",
     "RIGHTBORDER",   "LEFTBORDER",
            "TOPBORDER",      "BOTTOMBORDER",
     "TOGGLESELECT",  "STRINGCENTER",
            "STRINGRIGHT",    "LONGINT",
     "ALTKEYMAP",     "BOOLEXTEND"
  };
static char *systypenames[16] = {
     "SIZING",        "WDRAGGING",
            "SDRAGGING",      "WUPFRONT",
     "SUPFRONT",      "WDOWNBACK",
            "SDOWNBACK",      "CLOSE",
     NULL,            NULL,
            NULL,             NULL,
     "REQGADGET",     "GZZGADGET",
            "SCRGADGET",      "SYSGADET"
```

```
  };
static char *applitypenames[16] = {
     "BOOLGADGET",      "GADGET0002",
          "PROPGADGET",      "STRGADGET",
     NULL,              NULL,
          NULL,              NULL,
     NULL,              NULL,
          NULL,              NULL,
     "REQGADGET",       "GZZGADGET",
          "SCRGADGET",       "SYSGADGET"
  };
int sum, choice = -1;
USHORT bits;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
                    (APTR)gadget, DATASIZE, 0);
    switch (choice = GetChoice(DATASIZE)) {
     case 1:
      if (gadget->NextGadget)
        PrGadget("Gadget successivo nella lista
             di Intuition", gadget->NextGadget);
      break;
     case 6:
      bits = gadget->Flags;
      switch (bits & GADGHIGHBITS) {
        case 0:
          flagnames[0] = "GADGHCOMP";
          bits |= 0x01;
          break;
        case 1:
          flagnames[0] = "GADGHBOX";
          break;
        case 2:
          flagnames[1] = "GADGHIMAGE";
          break;
        case 3:
          flagnames[1] = "GADGHNONE";
          bits ^= 0x01;
          break;
      }
      FlagPrint("Flag di questo gadget",
             flagnames, (ULONG)bits);
      break;
     case 7:
      FlagPrint("Flag di attivazione di questo
             gadget",
        activatenames, (ULONG)gadget->Activation);
      break;
     case 8:
      bits = gadget->GadgetType;
      if (bits & SYSGADGET) {
        bits = (bits & 0xff00) | (1 << (((bits &
             0xf0) >> 4) - 1));
        FlagPrint("Flag che definiscono il tipo
             di questo gadget",
              systypenames, (ULONG)bits);
      }
      else {
        bits = (bits & 0xff00) | (1 << ((bits &
             0x0f) - 1));
        FlagPrint("Flag che definiscono il tipo
             di questo gadget",
          applitypenames, (ULONG)bits);
      }
      break;
     case 11:
      if (gadget->GadgetText)
        PrIntuiText("Prima struttura IntuiText
             nella lista",
                  gadget->GadgetText);
      break;
    }
  }
  level--;
}
```

Listato 5:

```
#include "sb.h"

extern int level;
extern void PrPlanes(), PrRastPort2();

void PrBitMap(string, bitmap) char *string; struct
BitMap *bitmap;
{
static struct StructData structdata[] = {
     { "-BytesPerRow", "UWORD",    12, INTSIZE    },
     { "-Rows",        "UWORD",    12, INTSIZE    },
     { "-Flags",       "UBYTE",    13, BYTESIZE   },
     { "-Depth",       "UBYTE",    13, BYTESIZE   },
     { "-Pad",         "UWORD",    12, INTSIZE    },
     { " Planes[8]",   "PLANEPTR", 5,  PTRSIZE * 8}
};
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
                      (APTR)bitmap, DATASIZE, 0);
    if ((choice = GetChoice(DATASIZE)) == 6)
      PrPlanes("BitPlane appartenenti a questa
             BitMap", bitmap->Planes,
           bitmap->Rows, bitmap->BytesPerRow);
  }
  level--;
}


void PrPlanes(string, planes, rows, bytes)
char *string; PLANEPTR planes[]; UWORD rows, bytes;
{
static struct StructData structdata[] = {
     {" BitPlane[0]",   "PLANEPTR", 5, PTRSIZE },
     {" BitPlane[1]",   "PLANEPTR", 5, PTRSIZE },
     {" BitPlane[2]",   "PLANEPTR", 5, PTRSIZE },
```

```
    {" BitPlane[3]",    "PLANEPTR", 5, PTRSIZE },
    {" BitPlane[4]",    "PLANEPTR", 5, PTRSIZE },
    {" BitPlane[5]",    "PLANEPTR", 5, PTRSIZE },
    {" BitPlane[6]",    "PLANEPTR", 5, PTRSIZE },
    {" BitPlane[7]",    "PLANEPTR", 5, PTRSIZE }
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
                (APTR)planes, DATASIZE, 0);
    choice = GetChoice(8);
    if (choice >= 1 && choice <= 8 && planes[choice
              - 1])
      HexDump(structdata[choice - 1].membername,
         planes[choice - 1], PTRSIZE, (long)(rows *
              bytes));
  }
  level--;
}


void PrRastPort(string, rastport) char *string;
struct RastPort *rastport;
{
static struct StructData structdata[] = {
  { "(Layer",      "struct Layer *)",     5, PTRSIZE  },
  { " BitMap",     "struct BitMap *",     5, PTRSIZE  },
  { "-Area Ptrn", "USHORT *",            5, PTRSIZE  },
  { "(TmpRas",     "struct TmpRas *)",    5, PTRSIZE  },
  { "(AreaInfo",   "struct AreaInfo *)", 5, PTRSIZE  },
  { "(GelsInfo",   "struct GelsInfo *)", 5, PTRSIZE  },
  { "-Mask",       "UBYTE",              13, BYTESIZE },
  { "-FgPen",      "BYTE",                3, BYTESIZE },
  { "-BgPen",      "BYTE",                3, BYTESIZE },
  { "-AOlPen",     "BYTE",                3, BYTESIZE },
  { "-DrawMode",   "BYTE",                3, BYTESIZE },
  { "-AreaPtSize","BYTE",                3, BYTESIZE },
  { "-linpatcnt",  "BYTE",                3, BYTESIZE },
  { "-dummy",      "BYTE",                3, BYTESIZE },
  { "(Flags",      "USHORT",             12, INTSIZE  },
  { "-LinePtrn",   "USHORT",             12, INTSIZE  }
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
              (APTR)rastport, DATASIZE, 0);
    switch (choice = GetChoice(MAXGADG + 1)) {
      case 2:
        if (rastport->BitMap)
          PrBitMap("BitMap di questa RastPort",
               rastport->BitMap);
        break;
      case MOREGADG:
        PrRastPort2("Membri della struttura
               RastPort (pagina 2)", rastport, sum);
        break;
    }
  }
  level--;
}


void PrRastPort2(string, rastport, offset)
char *string; struct RastPort *rastport; int offset;
{
static struct StructData structdata[] = {
  {"-cp_x",           "SHORT",           2, INTSIZE      },
  {"-cp_y",           "SHORT",           2, INTSIZE      },
  {" minterms[8]",    "UBYTE",           0, BYTESIZE * 8},
  {"-PenWidth",       "SHORT",           2, INTSIZE      },
  {"-PenHeight",      "SHORT",           2, INTSIZE      },
  {"(TextFont",       "struct Font *)",5, PTRSIZE      },
  {"-AlgoStyle",      "UBYTE",          13, BYTESIZE     },
  {"(TxFlags",        "UBYTE",          13, BYTESIZE     },
  {"-TxHeight",       "UWORD",          12, INTSIZE      },
  {"-TxWidth",        "UWORD",          12, INTSIZE      },
  {"-TxBaseLine",     "UWORD",          12, INTSIZE      },
  {"-TxSpacing",      "WORD",            2, INTSIZE      },
  {"-RP_User",        "WORD",            2, INTSIZE      },
  {" wordreserved[7]", "UWORD",          0, INTSIZE * 7 },
  {" longreserved[2]", "ULONG",          0, PTRSIZE * 2 },
  {" reserved[8]",    "UBYTE",           0, BYTESIZE * 8}
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
               (APTR)rastport, DATASIZE, offset);
    switch (choice = GetChoice(DATASIZE)) {
      case 3:
        HexDump("Stampa esadecimale dei byte di
              minterm",
            &rastport->minterms[0], BYTESIZE,
              (long)BYTESIZE * 8);
        break;
      case 14:
        HexDump("Stampa esadecimale di word
              riservate",
            &rastport->wordreserved[0],
              INTSIZE, (long)INTSIZE * 7);
        break;
      case 15:
        HexDump("Stampa esadecimale di longword
riservate",
            &rastport->longreserved[0],
PTRSIZE, (long)PTRSIZE * 2);
        break;
      case 16:
        HexDump("Stampa esadecimale di byte
riservati",
            &rastport->reserved[0], BYTESIZE,
(long)BYTESIZE * 8);
        break;
    }
  }
  level--;
}
```

Listato 6:

```
/* module written by G. Gagnon, Mar 24, 1987 */

#include "sb.h"

extern int level;


void PrIntuiText(string, intuitext) char *string;
struct IntuiText *intuitext;
{
static struct StructData structdata[] = {
   { "-FrontPen", "UBYTE",              3, BYTESIZE },
   { "-BackPen",  "UBYTE",              3, BYTESIZE },
   { "-DrawMode", "UBYTE",              3, INTSIZE  },
   { "-LeftEdge", "SHORT",              2, INTSIZE  },
   { "-TopEdge",  "SHORT",              2, INTSIZE  },
   { " ITextFont","struct TextAttr *",  5, PTRSIZE  },
   { " IText",    "UBYTE *",            4, PTRSIZE  },
   { " NextText", "struct IntuiText *", 5, PTRSIZE  }
   };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
           (APTR)intuitext, DATASIZE, 0);
    switch (choice = GetChoice(DATASIZE)) {
      case 6:
        if (intuitext->ITextFont)
          PrTextAttr("Struttura
             TextAttr",intuitext->ITextFont);
          break;
      case 7:
        PrString("Testo puntato da IText",
             intuitext->IText);
        break;
      case 8:
        if (intuitext->NextText)
          PrIntuiText("Struttura IntuiText
             successiva nella lista",
                  intuitext->NextText);
        break;
    }
  }
  level--;
}
```

Listato 7:

```
/* module written by G. Gagnon, Mar 24, 1987 */

#include "sb.h"

extern int level;

void PrMsgPort(string, msgport) char *string;
struct MsgPort *msgport;
{
static struct StructData structdata[] = {
   { "-mp_Node",       "Node structure",  0, 0        },
   { "   .ln_Succ",    "struct Node *",   5, PTRSIZE },
   { "   .ln_Pred",    "struct Node *",   5, PTRSIZE },
   { "-  .ln_Type",    "UBYTE",          13, BYTESIZE},
   { "-  .ln_Pri",     "BYTE",            3, BYTESIZE},
   { "   .ln_Name",    "CHAR *",          4, PTRSIZE },
   { "-mp_Flags",      "UBYTE",          13, BYTESIZE},
   { "-mp_SigBit"      "UBYTE",          13, BYTESIZE},
   { "(mp_SigTask",    "struct Task *)",  5, PTRSIZE },
   { "-mp_MsgList",    "List structure",  0, 0        },
   { "   .lh_Head",    "struct Node *",   5, PTRSIZE },
   { "   .lh_Tail",    "struct Node *",   5, PTRSIZE },
   { "   .lh_TailPred", "struct Node *",  5, PTRSIZE },
   { "-  .lh_Type",    "UBYTE",          13, BYTESIZE},
   { "-  .l_pad",      "UBYTE",          13, BYTESIZE}
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
           (APTR)msgport, DATASIZE, 0);
    switch (choice = GetChoice(DATASIZE)) {
      case 2:
        if (msgport->mp_Node.ln_Succ)
          PrNode("MsgPort->mp_Node.ln_Succ",
               msgport->mp_Node.ln_Succ);
        break;
      case 3:
        if (msgport->mp_Node.ln_Pred)
          PrNode("MsgPort->mp_Node.ln_Pred",
               msgport->mp_Node.ln_Pred);
        break;
      case 6:
        PrString("mp_Node->ln_Name",
               msgport->mp_Node.ln_Name);
        break;
      case 11:
        if (msgport->mp_MsgList.lh_Head)
          PrNode("MsgPort->mp_MsgList.lh_Head",
               msgport->mp_MsgList.lh_Head);
        break;
      case 12:
        if (msgport->mp_MsgList.lh_Tail)
          PrNode("MsgPort->mp_MsgList.lh_Tail",
               msgport->mp_MsgList.lh_Tail);
        break;
      case 13:
        if (msgport->mp_MsgList.lh_TailPred)
          PrNode("MsgPort->mp_MsgList.lh_TailPred",
               msgport->mp_MsgList.lh_TailPred);
        break;
    }
  }
  level--;
}
```

Listato 8:

```
/* module written by G. Gagnon, Mar 24, 1987 */

#include "sb.h"

extern int level;

void PrNode(string, node) char *string;
          struct Node *node;
{
static struct StructData structdata[] = {
   { " ln_Succ",    "struct Node *",    5, PTRSIZE  },
   { " ln_Pred",    "struct Node *",    5, PTRSIZE  },
   { "-ln_Type",    "UBYTE",            4, BYTESIZE },
   { "-ln_Pri",     "BYTE",             3, BYTESIZE },
   { " ln_Name",    "UBYTE *",          4, PTRSIZE  }
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
           (APTR)node, DATASIZE, 0);
    switch (choice = GetChoice(DATASIZE)) {
      case 1:
        if (node->ln_Succ)
          PrNode("Node->ln_Succ", node->ln_Succ);
          break;
      case 2:
        if (node->ln_Pred)
          PrNode("Node->ln_Pred", node->ln_Pred);
          break;
      case 5:
        PrString("ln_Name",node->ln_Name);
        break;
    }
  }
  level--;
}
```

Listato 9:

```
#include "sb.h"
extern int level;
extern void PrScreen2();

void PrScreen(string, screen) char *string;
          struct Screen *screen;
{
static struct StructData structdata[] = {
   { " NextScreen",  "struct Screen *", 5, PTRSIZE  },
   { " FirstWindow", "struct Window *", 5, PTRSIZE  },
   { "-LeftEdge",    "SHORT",           2, INTSIZE  },
   { "-TopEdge",     "SHORT",           2, INTSIZE  },
   { "-Width",       "SHORT",           2, INTSIZE  },
   { "-Height",      "SHORT",           2, INTSIZE  },
   { "-MouseY",      "SHORT",           2, INTSIZE  },
   { "-MouseX",      "SHORT",           2, INTSIZE  },
   { " Flags",       "USHORT",         12, INTSIZE  },
   { " Title",       "UBYTE *",         4, PTRSIZE  },
   { " DefaultTitle","UBYTE *",         4, PTRSIZE  },
   { "-BarHeight",   "BYTE",            3, BYTESIZE },
   { "-BarVBorder",  "BYTE",            3, BYTESIZE },
   { "-BarHBorder",  "BYTE",            3, BYTESIZE },
   { "-MenuVBorder", "BYTE",            3, BYTESIZE },
   { "-MenuHBorder", "BYTE",            3, BYTESIZE }
  };
static char *flagnames[8] = {
    "WBENCHSCREEN",     "CUSTOMSCREEN",
            NULL,               NULL,
    "SHOWTITLE",        "BEEPING",
            "CUSTOMBITMAP",     NULL
  };
int sum, choice = -1;
ULONG bits;
  level++;
  while (choice) {
  sum = SetOptionText(string, structdata,
                    (APTR)screen, DATASIZE, 0);
  switch (choice = GetChoice(MAXGADG + 1)) {
    case 1:
      if (screen->NextScreen)
        PrScreen("Screen successivo nella lista di
               Intuition", screen->NextScreen);
      break;
    case 2:
      if (screen->FirstWindow)
        PrWindow("Prima window dello screen",
               screen->FirstWindow);
      break;
    case 9:
      if ((bits = screen->Flags) & 2)
        bits ^= 1;
      FlagPrint("Flag dello screen", flagnames,
              bits);
      break;
    case 10:
      PrString("Titolo attuale dello screen",
              screen->Title);
      break;
    case 11:
      PrString("Titolo di default dello screen",
              screen->DefaultTitle);
      break;
    case MOREGADG:
      PrScreen2("Membri della struttura Screen
              (pagina 2)", screen, sum);
      break;
    }
  }
  level--;
}

void PrScreen2(string, screen, offset)
```

```
char *string; struct Screen *screen; int offset;
{
static struct StructData structdata[] = {
  { "-WBorTop",     "BYTE",              3, BYTESIZE },
  { "-WBorLeft",    "BYTE",              3, BYTESIZE },
  { "-WBorLeft",    "BYTE",              3, BYTESIZE },
  { "-WBorBottom",  "BYTE",              3, INTSIZE  },
  { " Font",        "struct TextAttr *", 5, PTRSIZE  },
  { "(ViewPort",   "struct ViewPort)", 0,SZ(ViewPort)},
  { " RastPort",   "struct RastPort ", 0,SZ(RastPort)},
  { " BitMap",      "struct BitMap",   0, SZ(BitMap) },
  { "(LayerInfo",   "struct Layer_Info)", 0,
                                      SZ(Layer_Info) },

  { " FirstGadget", "struct Gadget *",  5, PTRSIZE  },
  { "-DetailPen",   "UBYTE",           13, BYTESIZE },
  { "-BlockPen",    "UBYTE",           13, BYTESIZE },
  { "-SaveColor0",  "USHORT",          12, INTSIZE  },
  { "(BarLayer",    "struct Layer *)",  5, PTRSIZE  },
  { "(ExtData",     "UBYTE *)",         5, PTRSIZE  },
  { "(UserData",    "UBYTE *)",         5, PTRSIZE  },
 };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
              (APTR)screen, DATASIZE, offset);
    switch (choice = GetChoice(DATASIZE)) {
      case 5:
        if (screen->Font)
          PrTextAttr("Struttura
               TextAttr",screen->Font);
        break;
      case 7:
        PrRastPort("RastPort dello screen",
               &screen->RastPort);
        break;
      case 8:
        PrBitMap("BitMap dello screen",
               &screen->BitMap);
        break;
      case 10:
        if (screen->FirstGadget)
          PrGadget("Primo gadget dello screen",
               screen->FirstGadget);
        break;
    }
  }
  level--;
}
```

Listato 10:

```
/* module written by G. Gagnon, Mar 24, 1987 */

#include "sb.h"
extern int level;

void PrTextAttr(string, textattr) char *string;
         struct TextAttr *textattr;
{
static struct StructData structdata[] = {
  { " ta_Name",    "UBYTE *",     4, PTRSIZE  },
  { "-ta_YSize",   "UWORD",       2, INTSIZE  },
  { "-ta_Style",   "UBYTE",       3, BYTESIZE },
  { "-ta_Flags",   "UBYTE",       3, BYTESIZE }
 };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
          (APTR)textattr, DATASIZE, 0);
    switch (choice = GetChoice(DATASIZE)) {
      case 1:

        PrString("ta_Name",textattr->ta_Name);
        break;
    }
  }
  level--;
}
```

Listato 11:

```
#include "sb.h"
extern int level;
extern void PrWindow2(), PrWindow3();

void PrWindow(string, window) char *string;
         struct Window *window;
{
static struct StructData structdata[] = {
  { " NextWindow", "struct Window *",    5, PTRSIZE },
  { "-LeftEdge",   "SHORT",              2, INTSIZE },
  { "-TopEdge",    "SHORT",              2, INTSIZE },
  { "-Width",      "SHORT",              2, INTSIZE },
  { "-Height",     "SHORT",              2, INTSIZE },
  { "-MouseY",     "SHORT",              2, INTSIZE },
  { "-MouseX",     "SHORT",              2, INTSIZE },
  { "-MinWidth",   "SHORT",              2, INTSIZE },
  { "-MinHeight",  "SHORT",              2, INTSIZE },
  { "-MaxWidth",   "SHORT",              2, INTSIZE },
  { "-MaxHeight",  "SHORT",              2, INTSIZE },
  { " Flags",      "ULONG",             11, PTRSIZE },
  { "(MenuStrip",  "struct Menu *)",     5, PTRSIZE },
  { " Title",      "UBYTE *",            4, PTRSIZE },
  { "(FirstRequest", "struct Requester *)", 5,
                                         PTRSIZE },
  { "(DMRequest",  "struct Requester *)",5, PTRSIZE}
 };
static char *flagnames[32] = {
```

```
    "WINDOWSIZING",     "WINDOWDRAG",
            "WINDOWDEPTH",     "WINDOWCLOSE",
    "SIZEBRIGHT",       "SIZEBOTTOM",
            "SIMPLE_REFRESH", "OTHER_REFRESH",
    "BACKDROP",         "REPORTMOUSE",
            "GIMMEZEROZERO",   "BORDERLESS",
    "ACTIVATE",         "WINDOWACTIVE",
            "INREQUEST",       "MENUSTATE",
    "RMBTRAP",          "NOCAREREFRESH",
            NULL,              NULL,
    NULL,               NULL,
            NULL,              NULL,
    "WINDOWREFRESH",    "WBENCHWINDOW",
            "WINDOWTICKED"
  };
int sum, choice = -1;
ULONG bits;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
            (APTR)window, DATASIZE, 0);
    switch (choice = GetChoice(MAXGADG + 1)) {
      case 1:
        if (window->NextWindow)
          PrWindow("Window successiva nella lista
               di Intuition", window->NextWindow);
        break;
      case 12:
        bits = window->Flags & ~SUPER_UNUSED;
        switch ((bits & REFRESHBITS) >> 6) {
          case 0:
            flagnames[6] = "SMART_REFRESH";
            bits |= 0x40;
            break;
          case 1:
            flagnames[6] = "SIMPLE_REFRESH";
            break;
          case 2:
            flagnames[7] = "SUPER_BITMAP";
            break;
          case 3:
            flagnames[7] = "OTHER_REFRESH";
            bits ^= 0x40;
            break;
        }
        FlagPrint("Flag della window", flagnames,
               bits);
        break;
      case 14:
        PrString("Titolo della window",
               window->Title);
        break;
      case MOREGADG:
        PrWindow2("Membri della struttura Window
               (pagina 2)", window, sum);
        break;
    }
  }
  level--;
}

void PrWindow2(string, window, offset)
char *string; struct Window *window; int offset;
{
static struct StructData structdata[] = {
  { "-ReqCount",    "SHORT",               2, INTSIZE  },
  { " WScreen",     "struct Screen *",     5, PTRSIZE  },
  { " RPort",       "struct RastPort *",   5, PTRSIZE  },
  { "-BorderLeft",  "BYTE",                3, BYTESIZE },
  { "-BorderTop",   "BYTE",                3, BYTESIZE },
  { "-BorderRight", "BYTE",                3, BYTESIZE },
  { "-BorderBottom","BYTE",                3, BYTESIZE },
  { " BorderRPort", "struct RastPort *",5, PTRSIZE  },
  { " FirstGadget", "struct Gadget *",     5, PTRSIZE  },
  { " Parent",      "struct Window *",     5, PTRSIZE  },
  { " Descendant",  "struct Window *",     5, PTRSIZE  },
  { "(Pointer",     "USHORT *",            5, PTRSIZE  },
  { "-PtrHeight",   "BYTE",                3, BYTESIZE },
  { "-PtrWidth",    "BYTE",                3, BYTESIZE },
  { "-XOffset",     "BYTE",                3, BYTESIZE },
  { "-YOffset",     "BYTE",                3, BYTESIZE }
  };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
            (APTR)window, DATASIZE, offset);
    switch (choice = GetChoice(MAXGADG + 1)) {
      case 2:
        if (window->WScreen)
          PrScreen("Screen nel quale è contenuta la
               window",
             window->WScreen);
        break;
      case 3:
        if (window->RPort)
          PrRastPort("RastPort della window",
               window->RPort);
        break;
      case 8:
        if (window->BorderRPort)
          PrRastPort("BorderRPort della window",
               window->BorderRPort);
        break;
      case 9:
        if (window->FirstGadget)
          PrGadget("Primo gadget nella lista della
               window", window->FirstGadget);
        break;
      case 12:
        printf("Spiacente, non implementato\n\n");
        break;
      case 10:
        if (window->Parent)
          PrWindow("La window 'parente'",
               window->Parent);
        break;
      case 11:
        if (window->Descendant)
```

```
        PrWindow("La window 'discendente'",
                window->Descendant);
        break;
      case MOREGADG:
        PrWindow3("Membri della struttura Window
                (pagina 3)", window, sum);
        break;
    }
  }
  level--;
}


void PrWindow3(string, window, offset)
char *string; struct Window *window; int offset;
{
static struct StructData structdata[] = {
   { " IDCMPFlags", "ULONG",              1, PTRSIZE },
   { " UserPort", "struct MsgPort *",    5, PTRSIZE },
   { " WindowPort", "struct MsgPort *",  5, PTRSIZE },
   { "(MessageKey","struct IntuiMessage *)",5,PTRSIZE},
   { "-DetailPen", "UBYTE",             13, BYTESIZE},
   { "-BlockPen", "UBYTE",              13, BYTESIZE},
   { "(CheckMark", "struct Image *)",   5, PTRSIZE },
   { " ScreenTitle", "UBYTE",            4, PTRSIZE },
   { "-GZZMouseX", "SHORT",              2, INTSIZE },
   { "-GZZMouseY", "SHORT",              2, INTSIZE },
   { "-GZZWidth", "SHORT",               2, INTSIZE },
   { "-GZZHeight", "SHORT",              2, INTSIZE },
   { "(ExtData", "UBYTE *)",             5, PTRSIZE },
   { "(UserData", "BYTE *)",             5, PTRSIZE },
   { "(WLayer", "struct Layer *)",       5, PTRSIZE }
   };
static char *IDCMPnames[32] = {
     "SIZEVERIFY",      "NEWSIZE",
          "REFRESHWINDOW",   "MOUSEBUTTONS",
     "MOUSEMOVE",       "GADGETDOWN",
          "GADGETUP",        "REQSET",
     "MENUPICK",        "CLOSEWINDOW",
          "RAWKEY",          "REQVERIFY",
     "REQCLEAR",        "MENUVERIFY",
          "NEWPREFS",        "DISKINSERTED",
     "DISKREMOVED",     "WBENCHMESSAGE",
          "ACTIVATEWINDOW", "INACTIVEWINDOW",
     "DELTAMOVE",       "VANILLAKEY",
          "INTUITICKS",      NULL,
     NULL,              NULL,
          NULL,              NULL,
     NULL,              NULL,
          NULL,              "LONELYMESSAGE"
   };
int sum, choice = -1;
  level++;
  while (choice) {
    sum = SetOptionText(string, structdata,
            (APTR)window, DATASIZE, offset);
    switch (choice = GetChoice(DATASIZE)) {
      case 1:
        FlagPrint("Flag IDCMP abilitati in questa
                window",
                IDCMPnames, window->IDCMPFlags);
        break;
      case 2:
        if (window->UserPort)
          PrMsgPort("Window->UserPort",
                window->UserPort);
        break;
      case 3:
        if (window->WindowPort)
          PrMsgPort("Window->WindowPort",
                window->WindowPort);
        break;
      case 8:
        PrString("Titolo dello screen quando questa
                window è attiva",
                window->ScreenTitle);
        break;
    }
  }
  level--;
}
```

Sul prossimo numero:

- I device handler in dettaglio
- Analisi del compilatore C Lattice 5.0
- I cambiamenti operati nel filing system
- Come tenere sotto controllo la memoria di massa
- Indirizzamento indiretto tramite registro
- ARexx
- Il linguaggio Assembly su Amiga
- I dischi di Fred Fish
- IFF

Amiga Prosound Designer

Un digitalizzatore audio di qualità

È giunto anche sul mercato italiano, in verità già da un po' di tempo, la versione per Amiga del noto digitalizzatore audio Prosound Designer della ditta inglese Eidersoft.

Il prodotto, che era disponibile precedentemente solo per Atari ST, può essere acquistato in due forme distinte: hardware e software o solamente software. In un primo momento questo può sembrare un po' strano, visto che solitamente il software di controllo è relativo all'hardware specifico per cui è stato creato. Ciò, in questo caso, è smentito dal fatto che la Eidersoft ha incorporato nel proprio programma anche il supporto per unità di digitalizzazione di altri costruttori: FutureSound e PerfectSound. Nel caso che possediate uno di questi digitalizzatori, oppure uno di quei cloni piratati che sono disponibili in Italia, e non siate soddisfatti del software fornito a corredo, potete, da oggi, comprare il software Prosound Designer e mantenere lo hardware che avete. Il fatto di poter scegliere tra il software a corredo e un'alternativa come il prosound, non può che giovare.

Il manuale è stampato bene ed è anche ben organizzato, visto che guida l'utente passo dopo passo all'apprendimento di tutte le funzionalità messe a disposizione. Un difetto che si può riscontrare è quello di aver fornito un solo manuale che riguarda sia la versione ST che Amiga. Più spesso di quanto si creda, può capitare di essere tratti in inganno dal fatto che una spiegazione non si riferiva alla propria macchina. Comunque, un lettore attento, non può sbagliarsi poiché le sezioni relative a una macchina in particolare sono chiaramente evidenziate da un simbolo che non può lasciare dubbi, basta fare un po' di attenzione. Sul retro del manuale si trovano le foto degli autori delle diverse parti e versioni del Prosound Designer.

Il digitalizzatore vero e proprio è una piccola unità che si collega direttamente nella porta parallela dell'Amiga 1000 o, tramite un adattatore fornito, nella stessa porta del 500 e del 2000. Su un lato si trova l'ingresso per un mini-jack stereo. Sì, il digitalizzatore è stereo e il software supporta completamente sia il modo stereo che quello mono. I segnali audio possono essere mandati al digitalizzatore da virtualmente ogni sorgente sonora, ma probabilmente la più indicata è un Walkman stereo. Una piacevole sorpresa è il fatto che l'adattatore per la porta parallela del 500 e del 2000 è già fornito compreso nel prezzo: infatti se acquistato separatamente può anche costare più di 30.000 lire. Questo rende più appetibile il pacchetto visto che è pronto per essere utilizzato con qualunque Amiga in commercio.

Il software, per fortuna, non è assolutamente una conversione diretta dall'ST ma è stato scritto ex novo per fare uso delle superiori capacità sonore dell'Amiga. Dopo aver caricato il software da Workbench (veramente delle belle icone, ragazzi) nel modo usuale, l'utente passa attraverso una schermata iniziale corredata da suoni digitalizzati per poi giungere al pannello di controllo. A questo punto, data l'ottima qualità dell'interfaccia grafica, la somiglianza con altri software di digitalizzazione termina. È stato usato uno splendido effetto 3D per tutti i bottoni che servono per selezionare le diverse opzioni. Ciò fornisce al prodotto un ottimo aspetto professionale che non si può ritrovare in altri software analoghi per funzionalità.

Alla base dello schermo, che usa tutte le righe messe a disposizione dal PAL, si trovano i bottoni per la modifica della digitalizzazione corrente, che è mono oppure è uno dei due canali stereo. Le opzioni qui presenti prevedono la possibilità di sentire il brano (play), sentire il brano al rovescio (play backward), tagliare un sottobrano, cambiare la velocità di registrazione e di esecuzione, fondere o sovrapporre più parti. Infine sono presenti i selettori per il movimento dei puntatori di inizio/fine brano. Nello schermo di Prosound si trovano due aree grafiche: una grossa nella parte alta per il monitoraggio in tempo reale stereo dell'ingresso (utile per impostare a vista il livello di input) e un'altra stretta e lunga posta al centro che

mostra il campione digitalizzato attuale e che serve per compiere tutte le operazioni di editing appena menzionate. L'area contenente il campione è dotata di due puntatori che possono essere utilizzati per indicare una parte di brano su cui si possono poi eseguire una serie di operazioni: esecuzione, salvataggio e modifica.

Nella parte superiore sinistra dello schermo si trova un piccolo pannello che funge da archivio delle digitalizzazioni. I tasti, che sono numerati da F1 a F8, una volta selezionati con il mouse, mostrano un requester che permette di specificare quale campione caricare da disco. Quando la digitalizzazione è stata caricata il tasto viene, grazie a un effetto grafico basato su un uso accorto dei colori, illuminato, a indicare che è un tasto attivo. Selezionando un'altra volta con il mouse lo stesso tasto si sentirà il campione appena caricato da disco. Le digitalizzazioni possono anche essere ascoltate usando i tasti funzioni della tastiera. Eventualmente si possono attivare più digitalizzazioni allo stesso tempo e farle eseguire in un loop continuo, producendo così, alle volte, ottimi suoni del tipo FX. Se viene caricato un campione stereo, oppure due mono usati come un unico canale stereo, quando si seleziona uno dei tasti F per sentire una digitalizzazione, viene automaticamente attivato anche il canale associato. Si possono ottenere dei brillanti effetti di riverbero semplicemente spostando leggermente un puntatore di un canale rispetto a quello del canale stereo associato. Ogni canale stereo (destro o sinistro), può essere modificato indipendentemente dall'altro, portando, anche in questo caso, alla produzione di effetti sonori strabilianti. Il programma, forse non c'era bisogno di dirlo visto che dovrebbe essere uno standard, salva e carica campioni in formato IFF in modo che non si possano avere problemi durante lo scambio di dati con altri programmi compatibili IFF.

Diamo ora un'occhiata ad altre caratteristiche che non possono essere trovate in programmi di digitalizzazione paragonabili. La funzione zoom, per esempio, è eccellente, soprattutto perché lavora in modo cumulativo. Infatti l'utente può fare ingrandimenti e riduzioni come gli pare, fino ad arrivare a vedere gruppi di quattro byte. Se si effettua un ingrandimento (o più) e si posizionano i puntatori per indicare una certa porzione di digitalizzazione, quando si torna alla riduzione da cui si era partiti, i puntatori vengono riallineati nella posizione corretta sul grafico compresso. La possibilità di fondere più brani permette in pratica di realizzare un eco con numero di ripetizioni e a profondità variabile impostabili dall'utente; basta prendere, per mezzo dei puntatori, il pezzo di brano di cui si vuole fare l'eco e spostarlo verso destra fondendolo ogni tanto con il brano sottostante.

Nel terzo pannello visibile su schermo (come mai visibile su schermo? Leggete più avanti) posizionato nell'angolo superiore destro, si trovano dei bottoni che controllano la registrazione e l'esecuzione automatica dei campioni più un gadget denominato HI-FI. Quest'ultimo permette di disabilitare, sugli Amiga 500 e sui 2000 revisione B, il filtro audio a 7 KHz, rendendo così possibile l'ascolto alla massima qualità, di digitalizzazioni registrate a 14 KHz. Questa è una caratteristica non usuale che può realmente fornire qualità HI-FI, purché le digitalizzazioni siano più che buone.

Inoltre in quest'ultimo pannello si trova un tasto che, quando selezionato con il mouse, fa apparire un ulteriore pannello che scende dall'alto. Le possibilità che mette a disposizione l'ultimo arrivato sono diverse e tutte molto interessanti; a voi scoprire in che situazioni possono essere indispensabili. Per prima cosa si può aumentare o ridurre il volume di qualunque parte, o eventualmente tutto, di un campione, anche mentre lo si ascolta. Si può anche dimezzare o raddoppiare la frequenza di una parte o di tutta la digitalizzazione; utile per ridurre l'occupazione di memoria o per aumentare la qualità, oppure, naturalmente, per qualche effetto speciale di vostra invenzione. Ci sono anche due gadget per salvataggi veloci delle fasi di editing (per default verso il RAM disk).

Nel complesso le possibilità di modifica e trattamento dei campioni sono buone ma c'è sempre spazio per eventuali miglioramenti. Sicuramente abbiamo tralasciato poche delle possibilità più oscure di Prosound Designer, sia perché sono di difficile comprensione per i non esperti, sia perché saranno autentiche sorprese, che non vogliamo rovinare, per chi utilizzerà a fondo il software. Infatti, come per ogni buon prodotto, vi capiterà quasi sicuramente di ripetere: "Ehi, non sapevo che avrei potuto fare ciò in questo modo!". Prosound è proprio uno di questi prodotti.

Il software è stato scritto non dalla Eidersoft, ma da una ditta chiamata Digigraphic, che sta apparentemente lavorando ad altri progetti di Eidersoft, due dei quali sono relativi al Prosound. Uno di questi è il Prosound Toolkit che potrebbe essere pronto in tempi brevi: si tratta di una collezione di routine audio per i programmatori, più qualcos'altro per il miglioramento di digitalizzazioni e per conversioni particolari. Il secondo è una versione Gold (quindi qualcosa di veramente importante) del software Prosound. Non si hanno dati certi poiché viene mantenuto il più stretto riserbo sull'operazione.

Non si può che concludere dicendo che il programma ha delle ottime caratteristiche e fornisce anche delle prestazioni, per certi versi, uniche (evidentemente qualcuno ha passato molto tempo a farne un completo debug). Il software è veramente facile e divertente da usare e può essere veramente interessante per gli utilizzatori di altre unità di digitalizzazione. Già, il digitalizzatore, quasi ce ne stavamo dimenticando. Ha funzionato perfettamente e con diligenza senza provocare nessun problema. Fa quello che deve fare e ciò dovrebbe bastare. Per essere onesti preferiamo il digitalizzatore FutureSound quando si tratta di digitalizzare in mono, per via della presenza di più ingressi e per un controllo manuale del livello di ingresso. Sì, la vera mancanza del digitalizzatore Prosound (che, non dimentichiamolo, è stereo), è un potenziometro che regoli li livello di ingresso. Un'ultima nota riguardante il prezzo: basta guardarsi in giro per rendersi conto che per un prodotto originale, con garanzia e istruzioni (ahimè, in inglese), è veramente concorrenziale per quanto riguarda il prodotto completo, un po' meno per il solo software.

Il Prosound è acquistabile, in Italia, tramite SoftMail (via Napoleona 16, Como, tel. 031/300174) al prezzo di £ 139.000 per il solo software e £ 159.000 per il digitalizzatore più il software.

Difficoltà e scappatoie

Alcuni bug, alcuni pensieri e qualche idea

di Betty Clay

Esiste un computer senza bug? Ne dubito. Possibile che anche una piccola parte di software sia completamente libera da errori? Probabilmente no. I piccoli errori ci procurano molti problemi, e quando ne troviamo li correggiamo uno a uno, oppure scopriamo la maniera di aggirarli.

Questo articolo descrive alcune situazioni strane o fastidiose che sono state riportate recentemente. Parte di queste derivano da esperienze personali, ma la maggioranza ci è pervenuta da altre fonti. Le informazioni sono state raccolte da varie reti, riviste, notiziari e discussioni. Forse si riveleranno interessanti o utili, sta a voi decidere.

SETCOMMENT

Matt Dillon, autore di diversi programmi PD tra cui un famoso editor, ha riferito un problema avuto con SetComment nella versione 1.3. Sembra che Matt abbia introdotto un commento di dimensioni superiori a quelle permesse e il sistema operativo lo abbia copiato direttamente sul nome del file e sui campi seguenti del blocco di intestazione del file (file header block). L'operazione ha provocato un crash spettacolare, che si ripeteva a ogni riaccensione, e questo fino a quando non è stato eliminato il file rovinato.

Disfarsi di quel file non è comunque stata un'operazione semplice. Il nome del file era distrutto quindi il suo valore di hash era sbagliato e così pure il checksum. Avendo un valore di hash sbagliato era impossibile cancellare il file. Matt ha usato DISKED per disfarsi del file, ma una gran parte degli utenti Amiga non ha una conoscenza sufficiente per sbrigarsela da solo.

Ha dato un paio di suggerimenti alla Commodore: (1) SetComment non dovrebbe permettere di scrivere oltre la lunghezza prevista e (2) il DiskValidator non dovrebbe solo controllare le lunghezze impossibili dei nomi di file, ma dovrebbe anche accertarsi che il valore di hash e il checksum corrispondano. Questo è un problema che non possiamo correggere da soli, ma possiamo aggirarlo stando attenti alle dimensioni dei nostri commenti mentre attendiamo che la Commodore lo risolva.

Sempre Matt Dillon ha avuto un problema con i file di stampa dell'1.3 nel Native Developer's Kit. Ha scoperto che manca un XREF in ognuno di questi file, ma Matt dice che se cancellate quel riferimento quando appare il messaggio di errore, l'assembler può continuare senza di esso.

SPEAK:

Timothy Meisenheimer ha descritto un problema che ha avuto con il device SPEAK: che poi è stato confermato come errore, tuttavia si tratta di un bug facile da correggere.

Gli esempi dati riportano OPT in maiuscole, ma *deve* essere minuscolo per funzionare correttamente. Se solo tutte le soluzioni fossero così semplici!

RESIDENT

Il "bug" che si trova più frequentemente nella versione 1.3 è invece un errore di comprensione. Ci è stato segnalato da diverse persone e da parte di tutte le banche di raccolta dati che c'è un errore nel UseCount con il comando RESIDENT.

Per quante volte abbiate usato un comando, spesso mostra un UseCount uguale a 0.

UseCount segnala il numero di task che stanno in quel momento utilizzando un comando, non il numero di task che lo hanno usato dal momento dell'avvio. Se non si sta usando alcun task in quel momento, UseCount sarà quindi uguale a 0.

Dischi con Guru

Un altro problema che continua a causare noie è con i dischi che non si riescono a validare. Di solito appare un requester con la scritta: *Key is out of range* o *key is already set* o *key is already free*. Key corrisponde, naturalmente, al numero di settore dell'intestazione del file.

Se il key è già usato, il settore è considerato da un altro file come sua intestazione, e non possono esistere due file aventi lo stesso settore come intestazione. Se il key è troppo grande, significa che il sistema operativo ha scritto in un settore il cui numero è più grande di 1759 per un floppy, o più grande del

più alto numero di settore nella partizione di un hard disk. Ciò può essere corretto a mano se siete veramente capaci di usare un disk editor, ma di solito si può risolvere più facilmente con l'uso di DiskSalv.

Tale problema il più delle volte è causato da un difetto fisico del dischetto, ma può anche essere provocato da uno sbalzo di tensione della corrente o da qualcos'altro che disturba il drive durante la scrittura. Se vi succede spesso, state probabilmente usando dischi che sono vecchi e consunti, oppure di non ottima qualità sin dall'inizio o sono dischi a singola facciata. Nel caso dei floppy, l'uso di dischi migliori è la miglior soluzione.

Un altro disco rovinato ha causato un interessante crash sull'Amiga di Jim Butterfield. Jim ha provato a sostituire un file su un disco che era troppo pieno. Sembrava che l'operazione di registrazione fosse andata a buon fine, ma quando il disco è stato reinserito nel drive la volta successiva, tutto ha funzionato bene finché non è stata chiamata una particolare icona. Sullo schermo è esplosa una miriade di fuochi artificiali, dai chip del suono è uscito un lamento pietoso e la macchina si è bloccata.

Jim sapeva che a me piace studiare i dischi rovinati e con strani problemi, così me lo ha spedito. Esaminando il disco mi sono accorta che le dimensioni di quell'icona erano cresciute fino a 37K di lunghezza e alla fine del file dell'icona (.info file) ho trovato il file eseguibile.

In questo caso, ho copiato i file su un altro disco e ho salvato quello rovinato per ulteriori studi. DiskCopy non è utile in casi simili poiché copia sia i file sani che quelli contenenti un errore. Il comando Copy, invece, impiega molto più tempo, ma copia solo i file sani.

File nascosti-1

C'è un altro problema interessante dato da un disco che non ho ancora risolto: non ha dato un Guru e neppure altri segnali di allarme.

Durante la preparazione di un disco mensile per il mio Club stavo aggiungendo delle icone a tutti i file utilizzando un programma che aggiunge le icone automaticamente. Sul disco c'erano icone di directory di diversi tipi e colori e le ho cancellate tutte per poterle sostituire con icone che stessero bene.

Il programma si è rifiutato di fare nuove icone per le directory, allora ho provato a copiare una icona di directory a mano e non ci sono riuscita. Quindi ho aperto una directory e non ho più trovato nessun file, mi sembrò di aver perso il lavoro di tutta la giornata.

Ero disperata, ho provato a usare l'intero path per richiamare il programma e sono così riuscita a caricarlo e lanciarlo. Riuscivo a lanciare senza problemi tutti i programmi dei quali mi ricordavo il nome corretto, ma non c'era modo di vedere il loro nome né con il comando Dir né con il comando List.

Se chiamavo una subdirectory di cui mi ricordavo il nome, appariva normalmente. Ho risolto il problema immediato copiando (con il comando Copy e non DiskCopy) i file su un disco sano, usando "Copy DF0: to DF1: all". Sono così riuscita a copiare correttamente tutti i file su un nuovo disco dove si comportavano normalmente.

Non so ancora che cosa possa aver causato la perdita di tutti quei file sul disco originale. Così ho controllato i file originali con un disk editor per diverse volte, ma non ho ancora individuato niente di strano che possa aver causato questo comportamento. Forse qualcuno di voi potrà aiutarmi a dare una risposta a questo mistero.

File nascosti-2

Vi è mai successo di non riuscire a cancellare, cambiare il nome o comunque modificare un file sul disco che appare ancora nella directory? Ogni volta che richiamate la directory riuscite a vedere il file, ma qualsiasi cosa facciate non riuscite ad accedere al file.

La causa di questo è che il puntatore del file si trova nel posto sbagliato. Il puntatore di un file (il numero del blocco di intestazione) deve essere scritto nello slot della directory che corrisponde al valore di hash di quel nome del file. Se si trova in un blocco diverso il sistema operativo può ancora trovare il nome del file e metterlo in una directory, poiché legge il numero di settore dal blocco della directory, e poi ottiene il nome del file dal settore del blocco di intestazione.

Fare la cosa inversa è invece un'altra cosa. Quando richiamate un particolare file, il sistema prima ottiene il valore di hash del nome del file, poi cerca nello slot che corrisponde a quel valore il numero del settore dove trova il file.

Supponiamo che un nome di un file abbia valore di hash 12 (deve essere tra 6 e 78 compresi). Se il blocco di intestazione fosse #898, allora l'accesso alla directory sul disco sarebbe più o meno così:

```
slot    contenuto
0       002 tipo di file
1-10    ---
 11     xxx numero del blocco del file
            che ha  valore di hash 11
 12     898 numero del blocco di
                intestazione del nostro file
 13     xxx numero del blocco del file
                che ha  valore di hash 13
```

Dopo aver controllato lo slot numero 12, il sistema legge il blocco di intestazione 898, passa allo slot 107 (che è il byte # 432) e legge il nome del file. Se il numero del settore di intestazione del file si trova nello slot sbagliato, nello slot numero 12 ci sarà un'informazione sbagliata e non riuscirete a trovare il file.

Il numero potrebbe essere 0, il che indica che non esiste nessun file con quel nome; potrebbe trattarsi del numero del blocco dove inizia un altro file, e i nomi dei file non corrisponderanno. Il sistema controllerà l'intera catena di hash e alla fine si arren-

derà poiché non avrà trovato nessun file con il nome giusto.

In un caso come questo se riuscite a localizzare il file che cercate sul disco, potrete trovare il numero del suo blocco di intestazione all'inizio di ogni blocco e il valore di hash del nome del file, e poi mettere quel numero del blocco nello slot che corrisponde al nome del file. La maggior parte degli editor di dischi vi permettono di cercare una stringa sul disco. Potete quindi inserire il nome del file nella stringa da cercare e l'editor troverà il blocco di intestazione per voi.

RAD:

Molte persone stanno usando RAD: (Recoverable Ram Disk:, un RAM disk che non si cancella al reset del vostro Amiga) con soddisfazione, eseguendo lo startup dallo stesso ed usandolo come WorkBench in memoria. Altri, invece, pensano che crei molti problemi.

Nel mio Amiga c'è qualcosa che dà problemi con qualsiasi RAM disk resistente al reset. Penso che RAD: non sia molto utile perché ad ogni riavvio riporta un "Read-Write Error" e non si disfa dell'errore con un semplice riavvio. Mi dà più problemi che vantaggi.

Coloro che invece sono più soddisfatti del RAD: hanno chiesto come possono riavviare la macchina dal RAD: anche se hanno dimenticato di togliere il disco dal drive. Qualcuno sulla Usenet (penso fosse Andy Finkel) ha spiegato come fare.

Se settate il campo BootPri nella parte di descrizione del RAD: del file mountlist a 127, la vostra macchina si avvierà automaticamente da RAD: invece che da disco. Naturalmente a questo punto sarà impossibile avviare la macchina dal disco: non potete fare tutte e due le cose contemporaneamente. Beh, impossibile a meno che non usiate prima RemRad:.

DF1: per l'Amiga 2000?

Sul 2000 il drive interno è DF0:, ma un drive esterno sarà riconosciuto come DF2:. Molti programmi si aspettano di trovare i dischi in DF0: e in DF1: e non permettono altre configurazioni.

Questo non è gentile da parte dei programmatori, tuttavia l'utente ha bisogno di aggirare il problema. Se il file MountList nella directory devs: ha un elemento che dà il nome di DF1: ma ha 2 nell'UnitField (Unit=2) allora l'utente può aggiungere la linea "mount DF1:" al file Startup-Sequence e tutti i riferimenti al DF1: passeranno al device DF2: risolvendo tutti i problemi.

In altre parole il drive esterno risponderà sia a DF1: che a DF2:. Ciò rende felice il software. Nella versione 1.3 la MountList è già preparata e tutto quello che dovrete fare sarà aggiungere la linea alla Startup-Sequence.

Per coloro che ancora usano la versione 1.2 (c'è ancora qualcuno che lo fà?), il segno "/*" e "*/" dopo il segno "#" devono essere eliminati e bisogna scrivere UNIT=2 e HIGHCYL=79.

Sostituzione dei drive per il SideCard

Durante l'estate ho comprato un SideCard per usarlo con il mio Amiga 1000. Tuttavia le mie necessità di compatibilità con l'IBM sono estremamente limitate e il SideCard è diventato il regalo di natale per un parente molto speciale. Era molto felice e ha passato molto tempo adattandolo alle sue esigenze, ma il terzo giorno il drive ha smesso di funzionare. Abbiamo cercato di farlo riparare, ma nei pochi giorni in cui è rimasto ospite da me è stato tutto inutile.

Abbiamo deciso allora di comprare il pezzo di ricambio, ma non si poteva sostituire il drive rotto con un Chinon perché non erano compatibili. Così abbiamo comprato un Teac, ma anche questo non voleva funzionare. Altri amici con più esperienza vennero ad aiutarci, ma fu tutto inutile. Il SideCard era perfetto, e così pure il drive, ma sembrava che non riuscissero a comunicare.

Infine trovammo una soluzione molto semplice: mancava un ponticello tra RY e ML. Messo quello il drive funzionava perfettamente. Tutto è facile, una volta scoperto il trucco!

FONT Modificati

Geoffrey Kim ha segnalato su Usenet un altro bug. Dopo aver modificato il font Topaz 8, averlo salvato sotto altro nome nella directory Fonts: e aver cercato di usarlo con FastFonts si è trovato di fronte a un messaggio di errore che diceva che non poteva usare un font proporzionale, ma il suo font non era proporzionale!

Questo è successo a causa di un bug della versione 1.2 in FED (Font Editor). FED non ripuliva il bit ROMFONT nel campo dei flag settando comunque il flag del DISKFONT. FF (Fast Font) non accetta un font che abbia settati entrambi i flag.

È un po' confuso? Forse. Con la versione 1.2 del FED si può usare NewZap per correggere questo difetto, ma è più semplice procurarsi la versione 1.3. Pare che il nuovo FED abbia risolto il problema.

Il Virus Lazarus?

Numerose persone (di sicuro dei neofiti nel mondo Amiga) hanno recentemente segnalato sia su CompuServe che su Usenet un nuovo virus.

Hanno detto che il nuovo virus attaccava quando cercavano di recuperare un disco rovinato, sebbene stessero usando il metodo approvato dal software di sistema. Il disco si rovinava, il Requester diceva loro di usare il DiskDoctor per metterlo a posto e loro lo facevano.

Quando riottenevano il controllo del computer la struttura del disco era totalmente cambiata, alcuni dei file mancavano ed il nome del disco era cambiato in "Lazarus". Questo non è un virus, ma è una caratteristica del programma DiskDoctor.

Che il DiskDoctor qualche volta rinominasse un disco "Laza-

rus", mi fu chiaro nel 1985. DiskDoctor mette anche quasi tutti i file nella directory principale ed effettua altri cambiamenti che proibiscono ogni ulteriore tentativo di recupero.

E meglio non usarlo sebbene mi abbiano detto che la versione 1.3 sia molto più affidabile e dice molto di più su quanto sta facendo ai vostri file.

Il trucco di cambiare il nome del disco viene fatto quando DiskDoctor scopre che la directory principale ha dei problemi, specialmente se qualcosa impedisce la lettura del nome del disco. In questo caso viene dato al disco il nome "Lazarus" per indicare che è stato fatto resuscitare.

Malgrado questa operazione, quasi certamente il disco avrà seri problemi. Bisognerebbe copiare immediatamente i file su un altro disco e riformattare o buttare via il disco chiamato "Lazarus".

Il virus IRQ

Un altro virus, e questo è reale, è apparso all'inizio di quest'anno. Si tratta del virus IRQ e potrebbe essere il più tremendo di tutti quelli finora messi in circolazione. Questo virus è estremamente reale e cattivo.

Si attacca al primo comando chiamato dalla Startup-Sequence, istallandosi al posto del "load hunk" (ndt: la porzione del programma che è caricata per prima in memoria e descrive come deve essere caricato il resto) e mettendo il vecchio "load hunk" nel programma stesso.

Si manifesta cambiando il titolo della prima schermata con il suo nome, IRQ VIRUS.

Steve Tibbetts ha già fornito un killer per questo virus, ma sarà molto difficile evitare che infetti i nostri dischi continuamente. Si può attaccare a ogni programma eseguibile. Può trasmettersi in file trasferiti da e verso altri computer.

Il controllo dei file 1.3

Posso suggerire ai lettori di procurarsi, oltre al killer di Steve, una copia del SYSCHECK di Jim Butterfield? (Sarà sul disco corrispondente a questo numero di Transactor).

SysCheck leggerà il vostro disco WorkBench e vi darà il nome di tutti i file che sono diversi dalla versione 1.3.

Beh ... quasi tutti i file. Non controlla la Startup-Sequence, ma controlla, invece, tutti i comandi della directory C:, le Librerie, i Device, e altri file che difficilmente gli utenti si personalizzano.

Lanciando regolarmente SysCheck con il vostro WorkBench, individuerete i file che sono stati contaminati. Possono essere sostituiti con gli originali per ripulire la macchina. La combinazione di SysCheck e IRQKiller è la miglior protezione che fino ad oggi può essere usata.

Conclusioni

Un computer senza problemi né stranezze sarebbe molto meno interessante, almeno per me, di uno che ne ha poche. In qualche modo sembra che queste peculiarità diano una personalità alla macchina. Si scoprono in continuazione nuove difficoltà e trovare la soluzione a queste ci insegna di più sulla macchina. Avete notato che molti stanno ancora trovando strane cose nel Commodore 64 dopo tutti questi anni? Quanto sarà allora più interessante sfidare tutte le idiosincrasie dell'Amiga!

(segue da pagina 30)

Breakpoint

Non ho nulla in contrario allo smanettamento di un computer e al condurre esperimenti su di esso. La cosa alla quale faccio obiezione è il condurre gli esperimenti senza alcuna idea di quali *dovrebbero* essere i risultati.

Un altro problema creato dall'esecuzione passo passo di un programma senza alcuna idea di come dovrebbero essere i risultati, consiste nel fatto che veniamo indotti in una sorta di compiacimento dalla monotona correttezza della macchina e ci lasciamo sedurre credendo che una risposta sbagliata sia giusta.

"1 è un numero primo, 3 è primo, 5 è primo, 7 è primo....uffa, sembra proprio che tutti i numeri dispari siano primi. 9 è primo, 11 è primo, 13 è primo. YUP, tutti i numeri dispari *sono* primi."

Se quanto riportato sopra stesse accadendo in realtà e se un fisico stesse esaminando i risultati, si limiterebbe semplicemente a scartare il 9 come errore sperimentale. Ma i programmatori, oggigiorno, devono essere un po' più cauti. I computer vengono utilizzati in un numero sempre crescente di posti e di applicazioni. Prima o poi un errore software di un computer provocherà danni molto seri, magari ucciderà qualcuno! Facciamo in modo che *non* sia colpa del nostro programma.

I test per eliminare i bug da un software sono possibili, se attentamente studiati e implementati. Vediamo se riusciamo a individuare le trappole e a evitarle.

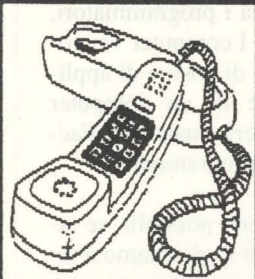
A proposito, quanti di voi hanno notato l'opinione dissenziente di Richard Feynman sul disastro del Challenger, in merito alle affermazioni ufficiali che la produzione e il test del software era l'*unica* cosa che sicuramente la NASA stava facendo correttamente?

Okay... la prossima volta daremo un'occhiata ai debugger più tradizionali disponibili su Amiga (a meno che, naturalmente, l'Avant-Garde e la Metadigm non rendano disponibili i loro nuovi prodotti nel frattempo) e proveremo a suggerire dei metodi per migliorare il *vostro* debugging.

SoftMail

Vendita per corrispondenza di programmi originali ed accessori per computers.

® SoftMail è un marchio registrato da Lago snc

Presentiamo in questa pagina alcune tra le ultime novità disponibili dal catalogo SoftMail. Ecco alcune informazioni utili per utilizzare il servizio SoftMail: è possibile effettuare ordini telefonicamente SOLO se è già stata effettuata una spedizione a proprio nome ed è stata regolarmente ritirata. Dal secondo in poi accettiamo ordini telefonici. Chi desidera notizie sulla disponibilità ed i prezzi dei prodotti che non compaiono in questa lista può chiamare lo (031)30.01.74 dalle 14:30 alle 18:00. SoftMail può organizzare la consegna anche tramite corriere: interpellateci per maggiori informazioni. Oltre alle ultime novità qui esposte, SoftMail offre l'intero catalogo delle seguenti case: Aegis, Cinemaware, EA, Microprose, Rainbird, SSI, SSG, Sublogic.

ACCESSORI

Articolo	Prezzo
Copritastiera A500	25.000
Final cartdrige III	110.000
Flicker master	29.000
Joy. Slik stick	16.500
Joy. SpeedKing	29.000
Joy. SpeedKing Autof.	33.000
Joy. Tac 2	29.000
Joy. Tac 5	39.000
MouseMat tappetino	22.500
Coprimouse	20.000
Portamouse	12.500
Portadischi 3" (30)	34.000
Portadischi 5" (40)	37.000
SlimLine (tastiera 64)	49.000
Voice messenger C64	60.000

LIBRI/HINTS & TIPS

Articolo	Prezzo
Alternate city clue specificare 8/16 bit	18.000
Bard's tale I clue	22.500
Bard's tale II clue	25.000
Bard's tale III clue	25.000
Black cauldron clue	18.000
Deathlord clue	telef.
Dungeon master clue	25.000
Elite clue	18.000
Graphic adv.creator hint	7.500
Mars saga clue	telef.
Might & Magic clue	25.000
Pool of radiance cle	20.000
Quest for clues	39.000
Sentinel world clue	25.000
Starflight clue	25.000
Ultima V clue	22.500
Wasteland clue	16.500

AMIGA

Articolo	Prezzo
Heroes lance hints disk	*telef.*
Bard's tale hints disk	*telef.*
Aegis draw 2000	telef.
Aegis images	69.000
Aegis impact	125.000
Aegis sonix	110.000
Aegis videotitiler 1.1	185.000
Alternate reality: the city	69.000
Animation multiplane	125.000
Armageddon man	49.000
Audiomaster	85.000
Audiomaster II	telef.
Baal	29.000
Barbarian II	telef.
Batman	29.000
Battlechess	49.000
Black cauldron	9.000
Blitzkrieg at Ardennes	59.000
California games	25.000
Carrier command	49.000
Cell animator	telef.
Chrono quest	65.000
Comic setter	139.000
Comic setter art disks	telef.
Corruption	49.000
Cosmic pirates	telef.
Crazy cars	29.000
Def con 5	59.000
Defender of the crown	59.000
Deluxe print II	99.000
DigiPaint	99.000
DigiView GOLD	telef.
Director	99.000
Disk drive esterno	299.000
Dos2Dos	75.000
Dragon's lair (1 MByte, 6 disks)	75.000
Driller TUTTO italiano	59.000
Dungeon master (1 MB)	59.000
Elite	45.000
Empire	49.000
F16 Falcon	59.000
Fantavision	125.000
Fire & forget	39.000
Fish	45.000
Flight simulator II	99.000
Scenery disks	telef.
Galileo 2.0	99.000
Garfield	49.000
Gauntlet II	25.000
Grabbit	49.000
Hellfire attack	39.000
International soccer	39.000
Italy 90 soccer	39.000
Incr.s.sphere	49.000
Jet & Japan bundle	110.000
Joan of Arc	25.000
King of Chicago	49.000
LED storm	25.000
Legend of the sword	45.000
Life cycles vol.1	39.000
Light!Camera!Action!	99.000
Lords of the Rising Sun	telef.
Major motion	39.000
Maxiplan plus	299.000
Menace	39.000
Mickey Mouse	25.000
Modeler 3D	129.000
Murder on Atlantic	59.000
Nigel Mansell grand prix	49.000
Obliterator	45.000
Offshore warrior	39.000
Outrun	25.000
P.O.W.	59.000
PacMania	25.000
Phantom fighter	45.000
Pioneer plague	59.000
President is missing	49.000
ProSound designer	79.000
ProSound w/hardware	199.000
Prowrite 2.0	179.000
Publisher plus	125.000
Reach for the stars	59.000
Rebel...Chickamauga	telef.
Return to Genesis	39.000
Rocket Ranger	59.000
Roger Rabbit	69.000
SEUCK	telef.
Sex vixens in space	55.000
Sculpt/Animate 4D	telef.
Sentinel	49.000
Sinbad	59.000
Skyfox II	59.000
Starfleet	59.000
Starglider II	49.000
Superman	39.000
Sword of sodan	69.000
The bard's tale II	49.000
The works!	249.000
Three Stooges	59.000
Tracker	49.000
TV Sports: Football	59.000
Ultima IV	49.000
Universal military sim.	45.000
Data disks 1 e 2	telef.
Victory road	39.000
Videoscape 3D 2.0	250.000
Designs disks	telef.
VIP professional	199.000
Virus	29.000
Virus infection protection	89.000
WB Extras	69.000
Whirligig	39.000
Willow	59.000
World Cl. Leaderboard	25.000
Zak McKracken	59.000
Zing keys	25.000
Zoetrope	199.000
Zoom	45.000
3 Demon	145.000

COMMODORE 64 CASS.

Articolo	Prezzo
After burner	18.000
Barbarian II	18.000
Batman	18.000
Dragon ninja	15.000
Exploding fist+	15.000
G.I.hero	15.000
Italy 90 soccer	18.000
Last Ninja II	25.000
MicroSoccer	telef.
Robocop	18.000
R-type	18.000
Serve & volley	22.000
Shoot'em up constr. kit	25.000
Stunt bike simulator	5.000
Tiger road	15.000
Total eclipse (ital.)	15.000
4 soccer simulator	18.000

COMMODORE 64 DISCO

Articolo	Prezzo
ADD: Heroes of the lance	telef.
ADD: Pool of radiance	59.000
American civil war III	49.000
Barbarian II	25.000
Deathlord	35.000
Driller	29.000
Echelon (con LipStick)	59.000
Eternal dagger	35.000
Fast break	29.000
Fish	29.000
Game over I+II	25.000
GeoPublisher	100.000
Grand Prix Circuit	telef.
Gulf strike	49.000
Home video producer	69.000
Italy 90 soccer	25.000
LED storm	15.000
Mars saga	35.000
McArhtu's war	49.000
MicroSoccer	telef.
Neuromancer	39.000
One on one II	29.000
Powerplay hockey	29.000
Red storm rising	49.000
Rocket Ranger	telef.
Shoot'em up constr. kit	35.000
Stealth mission	75.000
T.K.O.	29.000
The bard's tale III	35.000
Total eclipse (ital.)	20.000
Ultima V	49.000
Wasteland	39.000
Wec Les Mans	21.000
Who framed Roger Rabbit	telef.
Zak Mc Kracken	39.000
4 soccer simulator	25.000

COMMODORE 128 DISCO (128K, 80 COL.)

Articolo	Prezzo
"C" Language	99.000
Basic 8.0	75.000
GEOS 128	telef.
Mach 128	125.000

OFFERTE SPECIALI

(FINO AD ESAURIMENTO)

		Valore	A sole
FIREBIRD SPECIAL			
AMIGA	Bubble Bobble+Whirligig+ Enlightenment	107.000	75.000
C64 cass.	Savage+Intensity+Sentinel	48.000	33.000
C64 disco	Savage+Intensity+Sentinel	65.000	39.000
AEGIS ENTERTAINMENT			
AMIGA	Arazok's Tomb+Ports of Call	120.000	89.000

DEFENDER OF THE CROWN C64: cass. 12.000 disco 15.000

... SOFTMAIL TI DA' DI PIU'.

BUONO D'ORDINE da inviare a: LAGO DIVISIONE SOFTMAIL, VIA NAPOLEONA 16, 22100 COMO, FAX (031) 30.02.14, TEL (031) 30.01.74

Desidero ricevere i seguenti articoli:

- ☐ Addebitate l'importo sulla mia CARTASI/MASTERCARD/VISA/AMERICAN EXPRESS nr. ______ scadenza ______
- ☐ Pagherò al postino in contrassegno

Titolo del programma	Computer	Cassetta/disco/accessorio	Prezzo
		Contributo spese di spedizione Lit.	5.000
		TOTALE LIT.	

ORDINE MINIMO LIT. 20.000 (SPESE ESCLUSE)

Cognome e nome ______

Indirizzo ______ Nr. ______

CAP ______ Città ______ Prov. ______ Telefono ______

FIRMA (SE MINORENNE QUELLA DI UN GENITORE)

VERRANNO EVASI SOLO GLI ORDINI FIRMATI ______

I device di Amiga - 2

Come usare i device di Amiga nei vostri programmi

di Steve Simpson

Steve Simpson lavora come consulente programmatore per Amiga e IBM PC. Negli ultimi cinque anni ha vissuto in Svezia, lavorando come traduttore di documentazione tecnica. Si interessa attivamente dello sviluppo di utility DOS, di sistemi e del controllo di processi industriali; attualmente sta sviluppando alcune routine per il SideCar. Steve è anche l'autore di DiskRepair, una utility che recupera il contenuto di dischi danneggiati.

Chiunque abbia esplorato le possibilità offerte da Amiga nel campo dell'I/O conoscerà certamente i device di Amiga.

I device di Amiga si trovano in forma di file (con il suffisso .device) nella directory Devs sul disco di boot oppure sono già contenuti nella ROM del Kickstart. Il concetto di device è alla base dell'architettura di I/O "device independent" di Amiga (N.d.T.: "device independent" significa indipendente dal dispositivo con cui si intende colloquiare) e offre una forma di interfacciamento "omogeneo" tra il sistema e il mondo esterno.

Questo articolo è il secondo di una serie sull'argomento e deriva da un progetto al quale l'autore si è interessato; l'intento è di descrivere come un device di Amiga possa essere creato e aggiunto a quelli già presenti nel sistema. Dal momento che sussistono parecchie somiglianze tra device e library, può essere utile che prima leggiate l'articolo "Come creare delle run-time library", presente nel numero scorso, per avere un'idea di come viene implementata una library su Amiga.

Che cos'è un device?

Un device fornisce la possibilità di rendere le procedure di I/O indipendenti dal dispositivo interessato; il suo scopo è quello di fornire un interfacciamento standard tra sistema e mondo esterno e di rendere questo interfacciamento trasparente al sistema. Queste caratteristiche devono valere per tutti i dispositivi e per far questo è necessario che per ognuno di essi sia sempre disponibile un certo numero di funzioni e strutture standard, comuni a tutti i dispositivi.

Un dispositivo, d'altro canto, richiede anche routine specifiche e queste vanno opportunamente inserite nel contesto della definizione del device.

Questo è il meccanismo con il quale sono stati realizzati i device correntemente usati su Amiga, come il trackdisk.device, il console.device, l'input.device, il gameport.device e l'audio.device (che sono residenti in ROM) e il parallel.device, il serial.device, il printer.device e il clipboard.device (che invece vengono caricati in memoria da disco quando richiesti dal sistema).

La struttura di un device

Dal punto di vista della struttura, un device è molto simile a una run-time library di Amiga. La struttura Device è definita nel file "include" exec/libraries.h nella maniera seguente:

```
struct Device
{
struct Library dd_Library;
};
```

Così come una libreria, un device consiste di una tavola di indirizzi a cui saltare seguita da un nodo libreria e da informazioni varie, zona che è a sua volta seguita da un'area dati.

Il trackdisk.device è disposto nella seguente maniera (ricordatevi che gli indirizzi indicati potrebbero differire da quelli presenti nella vostra macchina):

```
offset
FFDC (-36)   JMP $00FEA06A   AbortIO
FFE2 (-30)   JMP $00FE9FBE   BeginIO
FFE8 (-24)   JMP $00FEA06A   ExtFunc
FFEE (-18)   JMP $00FE9F14   Expunge
FFF4 (-12)   JMP $00FE9F92   Close
FFFA (-6)    JMP $00FE9F42   Open
0000         Nodo library  - flags
                           - versione
                           - checksum
                           - numero di aperture
Area Dati
```

Ancora nello stesso modo delle library, la tavola cresce verso il basso nella memoria (a partire dal nodo library, in questo caso) e l'ordine dei suoi elementi rimane invariato; cambiano solo le

destinazioni dei salti, che dipendono da dove l'AmigaDOS ha caricato le routine del device.

Se il registro A6 contiene un puntatore alla struttura Device desiderata, l'istruzione assembler:

```
JSR      -30(A6)
```

comporterebbe l'esecuzione della funzione BeginIO. In assembler, l'offset può essere dichiarato come un valore globale nel file sorgente (con il prefisso _LVO) o definito come una costante nel file .i opportuno. Se _LVOBeginIO è definito come -30 ($FFE2), la funzione BeginIO può essere chiamata anche in questa maniera:

```
JSR      _LVOBeginIO(A6)
```

Se il device ha anche funzioni che possono essere chiamate senza passare attravero BeginIO, allora i loro offset possono essere ottenuti dalla posizione che le istruzioni di salto ad esse relative occupano dopo quelle relative alle funzioni standard del device. In assembler, questo si traduce nella definizione seguente:

```
LIBINIT
LIBFUNC     Dev_BeginIO
LIBFUNC     Dev_AbortIO
LIBFUNC     Dev_Func
ecc.
```

dove LIBINIT definisce gli offset relativi alle routine standard di una library (cioè quelli di Open, Close, Expunge e ExtFunc). Per saltare alla funzione Dev_Func si potrebbe dunque impiegare l'istruzione:

```
LINKLIB     Dev_Func,A6
```

dove, come al solito, supponiamo che A6 contenga il puntatore alla struttura Device.

Oltre alla consuete funzioni Open(), Close(), Expunge(), e ExtFunc(), che sono le routine che costituiscono il nocciolo di una library, in un device devono essere presenti due funzioni aggiuntive, BeginIO() e AbortIO().

BeginIO() è la routine di impiego generico per l'I/O presente in tutti i device; come ingresso richiede un puntatore alla struttura IORequest, che contiene le informazioni concernenti la richiesta di I/O in questione.

La definizione della struttura IORequest è presente nel file exec/io.h ed è qui riportata per comodità:

```
struct  IORequest
  {
  struct  Message io_Message;
  struct  Device  *io_Device;
  struct  Unit    *io_Unit;
  UWORD           io_Command;
  UBYTE           io_Flags;
  BYTE            io_Error;
  };
```

I membri della struttura hanno il seguente significato:

io_Message: contiene una struttura Message nella quale sono presenti informazioni associate alla struttura IORequest stessa che vengono usate sia per comunicare al device il completamento dell'I/O sia internamente per soddisfare secondo un ben preciso ordine le varie richieste.

io_Device: puntatore alla struttura Device associata alla richiesta di I/O in questione. Questo membro viene predisposto dal device all'apertura.

io_Unit: puntatore alla struttura Unit associata all'unità in questione del device. Anche questo membro è inizializzato dal device stesso all'apertura.

io_Command: il tipo di comando richiesto; può essere sia un un comando standard sia uno specifico del device.

io_Flags: impiegati per specificare particolari opzioni o per precisare lo stato del device.

io_Error: il numero d'errore restituito al completamento delle operazioni richieste.

I comandi possono essere sia standard e quindi comuni a tutti i device, sia peculiari del device. I comandi standard, come "read" e "write", vengono eseguiti da tutti i device per i quali un comando simile abbia senso, mentre i comandi specifici del device sono riconosciuti ed eseguiti solo da un ben particolare device; un esempio di quest'ultima classe di comandi è la richiesta di allocazione di un canale all'audio device.

I comandi standard sono i seguenti:

CMD_CLEAR: azzera tutti i buffer interni al device.

CMD_FLUSH: chiede al device di interrompere tutte le richieste di I/O in corso di esecuzione e quelle che sono ancora in attesa. A tutte le richieste viene risposto con un codice di errore.

CMD_INVALID: dice al device che il task che ha effettuato la richiesta ha mandato un comando non valido.

CMD_READ: richiede il trasferimento di un certo numero di byte di dati dai buffer interni al device a quello specificato dal programma chiamante.

CMD_RESET: riazzera il device (solo l'unità specificata) e le routine ad esso interne. Tutti i comandi in attesa di esecuzione sono persi.

CMD_STOP: blocca immediatamente il comando in corso di esecuzione. Le richieste che eventualmente arrivino nel frattempo sono poste in una lista di attesa.

CMD_UPDATE: richiede il trasferimento di tutti i dati presenti nei buffer interni al device al dispositivo fisico di I/O. Questo comando fa sì che non rimangano dati nei buffer interni che non siano ancora stati trasferiti a destinazione.

CMD_WRITE: trasferisce un certo numero di dati dal buffer del programma chiamante nei buffer del device.

Tutti i comandi standard sono individuati da valori definiti nel file exec/io.h; alla lista appena presentata si possono aggiungere i comandi specifici del device.

AbortIO() richiede come argomento un puntatore alla struttura IORequest e tenta di abortire dalle precedenti richieste di I/O. Questa funzione torna utile per cancellare anche tutte le richieste ancora in lista d'attesa.

È possibile aggiungere altre routine nella definizione del device; tali routine risulteranno associate al device. Per far questo si aggiungono, nella tabella che precede la struttura Device, ulteriori istruzioni di salto, oltre a quelle consuete per un device. Tali istruzioni devono ovviamente trasferire il controllo al codice relativo a tali routine aggiuntive. Un esempio è fornito dalla funzione RawKeyConvert(), che fa parte del console.device.

Le modalità operative

Un device può, all'apertura, lanciare uno o più task ed avere quindi uno o più "task control block" associati. Ogni task possiederà un proprio message port in modo da consentire il mantenimento in una lista di attesa delle richieste di I/O che gli pervengono. La struttura IORequest può essere vista come un tramite per inviare messaggi ai task che hanno la funzione di device. Non appena la richiesta associata a un messaggio viene soddisfatta, il device risponde al programma chiamante, restituendo la struttura IORequest. Alla luce di queste considerazioni, BeginIO(), SendIO() e DoIO() divengono nient'altro che applicazioni speciali del meccanismo di "message passing" di Amiga.

Il controllo dell'I/O mediante device può essere sia sincrono sia asincrono. La funzione BeginIO() realizza la modalità sincrona, che è anche la più semplice; BeginIO() viene chiamato con una struttura IORequest opportunamente inizializzata e non restituisce il controllo fino al completamento dell'operazione richiesta. Il programma chiamante sarà, cioè, posto in stato di attesa fino alla restituzione della struttura IORequest, che conterrà un codice d'errore oppure zero. Un device che operi in modalità sincrona non ha necessariamente dei task control block (relativi a sub-task) ad esso associati.

Il controllo asincrono richiede che il device abbia almeno un task con relativo message port associato. In tali condizioni, il device è in grado di offrire la possibilità di un controllo sia asincrono sia sincrono, con la gestione della lista di attesa dipendente dalla particolare implementazione del device. A seconda dello stato del task status flag del device, il messaggio relativo alla richiesta di I/O viene immediatamente inviato al task o inserito nella lista di attesa del message port. Il device prende in considerazione le richieste di I/O che così si accumulano col criterio FIFO, ovviamente. Il programma chiamante può allo stesso modo aspettare la risposta (I/O sincrono) o andare avanti e fare qualcos'altro nel frattempo (I/O asincrono). Un esempio di un task di device è quello dei task lanciati, uno per ogni unità del trackdisk.device (e di unità del trackdisk.device ce n'è una per ogni disk drive presente nel sistema).

Le Unità di un device

A ogni device possono essere associate diverse strutture Unit; tali strutture sono definite nel file exec/devices.h nella maniera seguente:

```
struct Unit
    {
    struct MsgPort   *unit_MsgPort;
    UBYTE            unit_Flags;
    UBYTE            unit_pad;
    UWORD            unit_OpenCnt;
    }
```

Il significato dei membri di questa struttura è il seguente:

unit_MsgPort: puntatore al message port del task associato all'unità.

unit_Flags: flag relativi alla lista d'attesa e al controllo del task.

unit_OpenCnt: numero di volte che questa unità è stata aperta.

La struttura Unit viene inizializzata dal device per ognuna delle unità aperte e un puntatore ad essa viene messo in un membro opportuno della struttura IORequest. Il numero dell'unità viene specificato come parametro quando si chiama la funzione OpenDevice(); tale numero può riferirsi a un'unità fisica o a un altro task di device. Per esempio, ogni disk drive viene considerato come un'unità a sé stante dal trackdisk.device; in parole più semplici, questo significa che è uno stesso programma ad occuparsi delle varie unità dischi, ma che ogni unità ha un proprio task control block, un proprio message port e proprie aree dati.

Altre strutture di richiesta di I/O

Tra quelli di sistema, solo i device audio, printer e timer impiegano la struttura IORequest prima descritta. Gli altri device usano una sua estensione, la struttura IOStdReq, definita nel file exec/io.h:

```
struct IOStdReq
    {
    struct  Message io_Message;
    struct  Device  *io_Device;
    struct  Unit    *io_Unit;
    UWORD           io_Command;
    UBYTE           io_Flags;
    BYTE            io_Error;
```

```
ULONG     io_Actual;
ULONG     io_Length;
APTR      io_Data;
ULONG     io_Offset;
ULONG     io_Reserved1;
ULONG     io_Reserved2;
};
```

io_Actual: il numero di byte effettivamente trasferiti nell'operazione di I/O richiesta.

io_Length: il numero di byte di cui è stato richiesto il trasferimento.

io_Data: puntatore al buffer dei dati del programma chiamante.

io_Offset: la spefica di allineamento (per i device, come il trackdisk.device, che la richiedono).

io_Reserved1: riservato per eventuali espansioni della struttura.

io_Reserved2: riservato per eventuali espansioni della struttura.

Un certo tipo di device può richiedere un'estensione della struttura IOStdReq o della IORequest. Ad esempio, il trackdisk.device usa una struttura IOExtTD per ospitare importanti informazioni aggiuntive:

```
struct  IOExtTD
{
struct IOStdReq     iotd_Req;
ULONG               iotd_Count;
ULONG               iotd_SecLabel;
};
```

in cui:

iotd_Req: è la struttura IOStdReq appena definita.

iotd_Count: conta il numero di estrazioni del disco.

iotd_SecLabel: è un puntatore a un buffer di 16 byte in cui vengono messe informazioni relative alla "sector label" del disco.

La struttura IOStdReq contiene al suo interno la IORequest come una sottostruttura ed è principalmente usata per restituire dati al programma chiamante; come regola di validità generale, si deve tener presente che il primo membro di una struttura, per così dire "personalizzata" per un device creato dall'utente, deve essere una struttura IORequest o IOStdReq.

Come inviare le richieste di I/O

Le richieste di I/O possono essere inviate ai device tramite le seguenti funzioni:

BeginIO(): invia al device la richiesta di I/O; il controllo può essere asincrono o sincrono, a seconda dell'implementazione del device.

DoIO(): invia la richiesta di I/O e attende il termine delle operazioni specificate prima di restituire il controllo al programma chiamante (caso sincrono).

SendIO(): invia la richiesta di I/O e restituisce immediatamente il controllo al programma chiamante, anche se non ha avuto luogo alcun trasferimento (caso asincrono).

AbortIO(): tenta di interrompere una richiesta di I/O fatta in precedenza.

WaitIO(): è usata per attendere il completamento delle operazioni associate a una richiesta asincrona di I/O. Il controllo non viene restituito sino al soddisfacimento della richiesta di I/O.

CheckIO(): è usata per controllare se una certa richiesta asincrona di I/O sia stata soddisfatta o meno.

DoIO(), SendIO(), WaitIO() e CheckIO() sono funzioni dell'Exec e pertanto fanno parte della exec.library, mentre BeginIO() è parte costitutiva della definizione del device; AbortIO(), invece, è definita tanto nel device quanto nella exec.library.

QuickIO

La modalità QuickIO rappresenta una scorciatoia che permette di evitare la lista di attesa che si forma al message port del device. Con il flag IOF_QUICK si fa presente al device che il programma vuole che l'I/O avvenga subito. Nel caso in cui ciò sia possibile, il device esegue quanto richiesto e comunica il risultato al programma chiamante. Sotto QuickIO il device non invia un messaggio di risposta al programma chiamante. Se un device non ha un proprio task allora la modalità QuickIO non comporta alcuna variazione rispetto alla situazione consueta. L'effettivo comportamento del device in modalità QuickIO dipende comunque dalla specifica implementazione del device stesso.

La funzione SendIO() si avvale del flag IOF_QUICK, ritornando al programma chiamante prima che la richiesta sia stata soddisfatta. DoIO(), al contrario, azzera tale flag e poi chiama WaitIO() per attendere il completamento dell'esecuzione della richiesta. Entrambe le funzioni comunque inviano la richiesta al device tramite la routine BeginIO() (che, lo ricordiamo, è parte integrante del device).

Ancora sulle Unit

Abbiamo visto che un'unità di un device è costituita da niente altro che un puntatore a un "task control block" (TCB) ovvero al message port di un task. A seconda del tipo di device, ci possono essere una o più unità e ognuna ha un TCB ad essa associato; in questa maniera ogni unità di un device consente la gestione di una lista di attesa di richieste di I/O, separata dalle altre. La possibilità di avere più unità è ovviamente sfruttata a livello di sistema: il timer device ha, infatti, due unità, l'audio

device ne ha quattro e il console device può invece averne un numero illimitato.

In definitiva, tutte le richieste, tanto sincrone quanto asincrone, giungono all'unità attraverso la funzione BeginIO(); questa funzione può chiamare direttamente la funzione ExecuteIO(), se è richiesta la modalità QuickIO, o può inserire la richiesta nella lista di attesa del message port dell'unità prescelta.

Il task relativo all'unità, quando rimuove una richiesta di I/O dalla lista del message port, passa alla routine ExecuteIO() il puntatore alla struttura IORequest rimossa e un puntatore alla struttura Device interessata. A questo punto, il task alza un flag che fa in modo che altre eventuali richieste siano poste in lista d'attesa fino al completamento della richiesta in corso di esecuzione. La funzione ExecuteIO() individua l'indirizzo della funzione associata al comando nella tabella dei comandi (cmdTable) e salta a tale locazione; dopodiché, quello che accade dipende strettamente dal tipo di device, ovviamente.

Al termine dell'esecuzione del comando, il controllo viene passato a TerminateIO(), che azzera i flag della unità e rispedisce la struttura IORequest al programma che l'aveva inviata, con una chiamata alla funzione ReplyMsg().

Le routine di un device

Le routine presenti in un device si dividono in due categorie principali: i comandi, cioè le routine che vengono eseguite in risposta a una richiesta di I/O, e le funzioni, alle quali si accede, come per le funzioni di una library, attraverso un offset relativo all'indirizzo base della struttura Device. Le funzioni e i comandi possono essere scritti indifferentemente in assembler o in C; devono, comunque, essere dichiarati come oggetti esterni (in assembler, con XREF <nome funzione>) nel sorgente della parte principale del device, in modo che possano essere individuati correttamente dal linker.

Ai comandi, che poi sono le routine che effettuano le vere e proprie operazioni di I/O, è associato un puntatore nella tavola dei comandi del device. Il membro io_Command della struttura IORequest specifica quale comando eseguire. Generalmente i parametri vengono passati alle routine attraverso i registri e queste si occupano di aggiornare opportunamente i campi io_Actual e io_Error della struttura IORequest prima di restituire il controllo.

Nel caso in cui voleste scrivere dei comandi in C, ricordatevi che è necessario andare a pescare i parametri nei registri e trasferirli nell'ordine corretto sullo stack dove una routine scritta in C si aspetta di trovarli.

Le funzioni possono essere usate per predisporre delle modalità di funzionamento del device o per usufruire di particolari possibilità messe a disposizione da un certo device (come, ad es., RawKeyConvert() nel console.device). Alle funzioni di un device si accede nella stessa maniera in cui si accede alle funzioni di una library; per poterle implementare in C, è necessario servirsi di una "interfaccia" (finora obbligatoriamente scritta in assembler, ma che, con alcune nuove release dei compilatori, potrà essere scritta direttamente in C), che si occupi di trasferire i parametri dai registri sullo stack nel giusto ordine e che poi chiami la routine C che costituisce la funzione vera e propria. Inoltre, per accedere a tali funzioni da C, risulta necessario scrivere un'altra interfaccia, usata al momento del linking, individuata dal nome della funzione in questione e che ha il compito di passare il controllo all'istruzione di salto corrispondente alla funzione prescelta nella jump-table del device.

Come allestire un device

Le varie parti che vanno a comporre un device vengono compilate senza il consueto modulo di startup (.begin per chi usa il Manx e c.o per chi usa il Lattice C) dal momento che esso risulta inappropriato per l'uso che intendiamo fare del codice. Durante la compilazione o l'uso dell'assembler, è opportuno impiegare le opzioni "large code" e "large data"; inoltre, è buona norma che il nome del file che costituisce il device abbia il suffisso ".device".

La struttura Device che abbiamo prima considerato può essere estesa. Nell'esempio di device che segue l'articolo, alla struttura Device facciamo seguire un certo numero di puntatori e di flag che sono frequentemente usati nel contesto del device. In C, la struttura usata nel device di esempio potrebbe essere così definita:

```
struct  MyDev
   {
   struct  Device     md_Device;
   struct  ExecBase   *md_ExecBase;
   struct  DosBase    *md_DosBase;
   ULONG              md_SegList;
   UBYTE              md_Flags;
   UBYTE              md_pad;
   struct  Unit       *md_units[MD_NUMUNITS];
   };
```

md_Device: la struttura Device associata al device.

md_ExecBase: puntatore alla exec.library.

md_DosBase: puntatore alla dos.library.

md_SegList: puntatore BCPL alla "segment list" del device.

md_Flags: flag impiegati nel device.

md_units: array di puntatori alle unità del device.

Il nostro esempio di device

Il ROM Kernel Manual: Libraries and Devices (RKM) presenta un esempio di come creare un device, ma gli svariati errori che compaiono nel listato rendono tale esempio virtualmente inutile ai fini pratici.

Il listato mydev.asm (listato 3), insieme con i moduli di supporto mydev_def.i (listato 1) e asmsupp.i (listato 2), fornisce il codice per un device (device che questa volta, invece, funzio-

na!). L'esempio del RKM, una volta individuati gli errori che lo costellano, è servito solo come traccia.

Una gran parte dei dettagli implementativi di un device sono identici a quelli di una run-time library; c'è, comunque, un'importante differenza nella "function table" o funcTable, come è chiamata nel file mydev.asm, che contiene gli indirizzi assoluti delle routine del device. L'ordine di tali indirizzi è importante; infatti, gli indirizzi delle funzioni scritte dall'utente devono apparire dopo l'indirizzo relativo alla funzione AbortIO(). La tavola è terminata da un -1L.

L'interfaccia con il sistema

Un device di Amiga deve comprendere un insieme minimo di 6 funzioni standard: Open(), Close(), Expunge(), ExtFunc(), BeginIO() e AbortIO(). I riferimenti a OpenDevice(), CloseDevice() e RemDevice() vengono tradotti ripspettivamente in chiamate a Open(), Close() e RemDevice() mentre un riferimento a BeginIO(), DoIO() o SendIO() coinvolge la funzione BeginIO definita nel device.

DoIO(), SendIO() e BeginIO() sono funzioni definite nella solita link library (amiga.lib); BeginIO(), in particolare, non è altro che un semplice modulo di interfaccia alla routine BeginIO del device.

Come installare un device

Il listato 4, add_dev.c, illustra come procedere per rendere disponibile al sistema un device. Il device viene caricato in memoria da LoadSeg(), dopodiché una chiamata a MakeLibrary() inizializza la jump-table del device, predispone le aree dati ed esegue le routine di inizializzazione eventualmente fornite dall'utente.

Gli argomenti della funzione MakeLibrary() comprendono un puntatore alla tabella degli indirizzi delle routine del device e alla tabella di inizializzazione dei dati, un puntatore alla routine di inizializzazione del device, il numero di byte richiesti dal device, nonché il puntatore BCPL restituito da LoadSeg(). Il numero di byte di memoria richiesto è quello che serve per alloggiare la struttura del device e la zona dati.

Con l'istruzione

```
AddDevice(myDevice);
```

si aggiunge la struttura Device puntata da myDevice nella lista di device gestita da Exec, rendendo in questa maniera disponibile il nuovo device ai vari programmi applicativi che possono averne bisogno. C'è anche un altro mezzo per installare un device nel sistema. Si può salvare il device nella directory logica DEVS:, cioè dove il sistema va a vedere quando non trova nelle liste di sistema il device richiesto con OpenDevice(). Se il device richiesto viene trovato, il sistema operativo lo carica in memoria e lo inizializza automaticamente; affinchè tutto funzioni correttamente, però, il device deve essere strutturato nella maniera illustrata nel listato mydev.asm.

L'impiego di un device

È necessario seguire una certa procedura prima di accedere alle funzioni e ai comandi messi a disposizione da un device.

Prima di richiedere l'apertura del device, infatti, bisogna predisporre un message port attraverso il quale avverrà il dialogo con il device stesso e allocare la struttura IOStdReq o una sua estensione. Vediamo in maggior dettaglio i passi necessari per l'uso di un device.

- Il device abbisogna di un message port per restituire le strutture di richiesta I/O con i codici d'errore o altre utili informazioni. Per creare questo port potete far ricorso alla funzione CreatePort() definita in amiga.lib.
- Il device può impiegare la struttura IORequest o IOStdReq oppure una specifica per un particolare uso. Nel caso in cui si usino le strutture standard, ci si può rivolgere a funzioni definite nella link library amiga.lib, che sono CreateStdIO() e CreateExtIO() e che si occupano di allocare ed inizializzare correttamente tali strutture.
- Per avere accesso alle funzioni e ai comandi si può ora chiamare OpenDevice(). Si ricordi che un device può consentire che coesistano più unità distinte.
- Al termine del programma che usa il device, non ci si dimentichi di chiamare CloseDevice(), di liberare tutta la memoria allocata per le strutture relative all'I/O e di rilasciare il message port. A tale scopo si usano le funzioni DeleteStdIO(), DeleteExtIO() e DeletePort().

Il listato 5 (main.c) mostra come procedere per eseguire correttamente le inizializzazioni richieste; in particolare, ci sono esempi di invio di richieste di I/O e di chiamate alle funzioni del device. Nel modulo di interfaccia devfunc.asm (listato 6) si illustra nel dettaglio come accedere alle funzioni del device.

Uso dei registri

Un certo numero di registri sono convenzionalmente riservati per scopi particolari. Il registro A6 contiene un puntatore alla struttura Device (o Library), mentre A0 e A1 sono generalmente usati per passare gli argomenti a una funzione, il cui risultato è poi posto in D0 e D1. A0, A1, D0 e D1 possono essere considerati come registri di lavoro e il loro contenuto può non essere conservato durante l'esecuzione delle routine.

Conclusione

Tutte le richieste di I/O sono trattate dal sistema fondamentalmente nella stessa maniera; una tale indipendenza del metodo dal dispositivo interessato è uno dei punti di forza dell'approccio di Amiga all'I/O.

Questo articolo ha tentato di fare un po' di luce su quell' area mal documentata dell'ambiente di Amiga che è l'implementazione dei device e di chiarire come sia possibile un interfacciamento formalmente uniforme con un'ampia varietà di

dispositivi, dalle tavolette digitalizzatrici ai plotter e alle stampanti laser.

I programmi allegati a questo articolo sono stati sviluppati con il compilatore Manx Aztec C68k v3.60a e il debugger SDB.

Listato 1: mydev_def.i

```
;---------------------------------------------------
; mydev_def.i
;
; definizioni assembler per mydev.device
;
; Storia:
;    88-05-16  creato
;---------------------------------------------------

;offset delle funzioni del device rispetto alla base
;della struttura
 LIBINIT
  LIBDEF    Dev_BeginIO
  LIBDEF    Dev_AbortIO
  LIBDEF    Dev_Func


MD_NUMUNITS   EQU   4

;strutture dell'area dati del device mydev
 STRUCTURE   MyDev,LIB_SIZE
   ULONG  md_ExecBase
   ULONG  md_DosBase
   ULONG  md_SegList
   UBYTE  md_Flags
   UBYTE  md_pad
   STRUCT md_Units,MD_NUMUNITS*4
   LABEL  MyDev_Sizeof

 STRUCTURE MyDevMsg,MN_SIZE
   APTR   mdm_Device
   APTR   mdm_Unit
   APTR   mdm_Iob
   LABEL  MyDevMsg_Sizeof

 STRUCTURE MyDevUnit,UNIT_SIZE
   UBYTE  mdu_UnitNum
   UBYTE  mdu_pad
   STRUCT mdu_Msg,MyDevMsg_Sizeof
   APTR   mdu_Process
   LABEL  MyDevUnit_Sizeof

;flag di stato per le unit fermate
 BITDEF  MDU,STOPPED,2

;dimensione dello stack e priorita' per il processo
;che creeremo

MYPROCSTACKSIZE  EQU    $400
MYPROCPRI        EQU    0

;nome del mio device
MYDEVNAME            MACRO
      dc.b      'mydev.device',0
      ENDM
```

Listato 2: asmsupp.i

```
;---------------------------------------------------
; asmsupp.i
;
; macro addizionali
;
; Storia:
;    88-05-06  creato
;---------------------------------------------------

CLEAR            MACRO
    moveq    #0,\1
    ENDM

LINKSYS          MACRO
    LINKLIB  _LVO\1,\2
    ENDM

CALLSYS          MACRO
    CALLLIB  _LVO\1
    ENDM

XLIB             MACRO
    XREF     _LVO\1
    ENDM
```

Listato 3: mydev.asm

```
;---------------------------------------------------
;
; mydev.data -- esempio del codice del device
;
;Esempio di come si possa costruire un device di
;sistema
;
; Storia:
;    88-05-17  creato
;    88-11-06  versione finale
;
; assemblaggio:  as -c -d -l -isys2:asm mydev
; link:  ln -o mydev.device mydev sys3:lib/cl.lib
;
```

```
; Nota: usate il modello large code e large data
;       senza modulo di startup
;       (.begin per Manx, startup.obj per Lattice)
;
;---------------------------------------------------

 SECTION     section

 NOLIST
 INCLUDE "exec/types.i"
 INCLUDE "exec/nodes.i"
 INCLUDE "exec/lists.i"
 INCLUDE "exec/alerts.i"
 INCLUDE "exec/resident.i"
 INCLUDE "exec/initializers.i"
 INCLUDE "exec/memory.i"
 INCLUDE "exec/libraries.i"
 INCLUDE "exec/devices.i"
 INCLUDE "exec/io.i"
 INCLUDE "exec/ables.i"
 INCLUDE "exec/errors.i"
 INCLUDE "libraries/dos.i"
 INCLUDE "libraries/dosextens.i"
 INCLUDE "asmsupp.i"
 INCLUDE "mydev_def.i"

 LIST


;---------------------------------------------------
; definizioni e riferimenti esterni
;---------------------------------------------------

XDEF     DevInit
 XDEF     Open
 XDEF     Close
 XDEF     Expunge
 XDEF     ExtFunc
 XDEF     myName
 XDEF     BeginIO
 XDEF     AbortIO
 XDEF     DevFunc

 XREF     _AbsExecBase

 XLIB     OpenLibrary
 XLIB     CloseLibrary
 XLIB     Alert
 XLIB     FreeMem
 XLIB     Remove
 XLIB     FindTask
 XLIB     AllocMem
 XLIB     CreateProc
 XLIB     PutMsg
 XLIB     RemTask
 XLIB     ReplyMsg
 XLIB     Signal
 XLIB     GetMsg
 XLIB     WaitPort
 XLIB     AllocSignal
 XLIB     SetTaskPri

 INT_ABLES

 DEVINIT
  DEVCMD    MYDEVCMD
  DEVCMD    MYDEV_END


;---------------------------------------------------
;La prima locazione eseguibile. Restituisce un
;errore nel caso che qualcuno cerchi di eseguire il
;device come se fosse un preogramma
;---------------------------------------------------

DevStart:
         CLEAR    d0
         rts


;---------------------------------------------------
; definizioni ed EQU varie
;---------------------------------------------------

    ;priorita' della libreria
MYPRI    EQU      0
    ;major number della versione
VERSION  EQU      1
    ;una revisione particolare. Identifica in modo
    ;univoco la libreria
REVISION EQU      1


;---------------------------------------------------
;Una struttura romtag.
;'exec' e 'ramlib' ricercano questa struttura
;durante la fase di caricamento per scoprire, per
;esempio, da quale punto debba partire l'esecuzione.
;Cio' e' equivalente alla struttura C Resident.
;---------------------------------------------------

initDevDescrip:         ;stuttura RT,0
         dc.w     RTC_MATCHWORD    ;UWORD
RT_MATCHWORD
         dc.l     initDevDescrip   ;APTR  RT_MATCHTAG
         dc.l     EndDevCode       ;APTR  RT_ENDSKIP
         dc.b     RTF_AJTOINIT     ;UBYTE RT_FLAGS
         dc.b     VERSION          ;UBYTE RT_VERSION
         dc.b     NT_DEVICE        ;UBYTE RT_TYPE
         dc.b     MYPRI            ;UBYTE RT_PRI
         dc.l     myName           ;APTR  RT_NAME
         dc.l     DevIdString      ;APTR  RT_IDSTRING
         dc.l     DevInit          ;APRT  RT_INIT
                                   ;LABEL RT_SIZE

    ;nome del mio device
myName          MYDEVNAME
```

```
;--------------------------------------------------
;Identificatore del device tag usato per supportare
;il device.
;Il formato e':
;          'nome versione.revisione (dd mmm yyyy)',
           <cr>,<lf>,<null>
;--------------------------------------------------

DevIdString    dc.b     'mydev 1.0 (11 August
1988)',13,10,0

    ;nome della dos library da aprire
dosName        DOSNAME

    ;forza l'allineamento su word
        ds.w     0


;--------------------------------------------------
;romtag specifica "RTF_AUTOINIT". Il campo RT_INIT
;punta alla tabella seguente. Se RTF_AUTOINIT non e'
;impostato, allora RT_INIT punta a una routine di
;inizialilzzazione che deve essere eseguita.
;Questi dati sono usati dal programma di caricamento
;come parametri per MakeLibrary()
;--------------------------------------------------

DevInit:
         dc.l     MyDev_Sizeof  ;dimensione dei dati
         dc.l     funcTable     ;puntatore agli
                                ;inizializzatori
                                ;delle funzioni
         dc.l     dataTable     ;puntatore agli
                                ;inizializzatori
                                ;dei dati
         dc.l     initRoutine   ;routine da
                                ;eseguire

;--------------------------------------------------
;Tabella degli indirizzi delle funzioni mydev
;--------------------------------------------------

funcTable:
    ;routine standard di sistema - devono essere
    ;sempre incluse
         dc.l     Open
         dc.l     Close
         dc.l     Expunge
         dc.l     ExtFunc
    ;definizioni per il mio device
         dc.l     BeginIO
         dc.l     AbortIO
    ;funzioni del mio device
         dc.l     DevFunc
    ;marcatore di fine tabella
         dc.l     -1
```

```
;--------------------------------------------------
;Questa tabella inizializza le strutture dati
;statiche. Il formato e' specificato nelle pagine
;del manuale RKM relative a exec/InitStruct. Il
;primo argomento e' l'offset rispetto alla base
;della libreria del device. Il secondo e' il valore
;da porre in quella cella. La tabella e' terminata
;con un NULL.
;--------------------------------------------------

dataTable:
         INITBYTE     LN_TYPE,NT_DEVICE
         INITLONG     LN_NAME,myName
         INITBYTE     LIB_FLAGS,
                      LIBF_SUMUSED!LIBF_CHANGED
         INITWORD     LIB_VERSION,VERSION
         INITWORD     LIB_REVISION,REVISION
         INITLONG     LIB_IDSTRING,DevIdString
         dc.l         0


;--------------------------------------------------
;Questa e' la routine di inizializzazione - viene
;chiamata dopo che il device e' stato allocato.
;Se restituisce un valore diverso da zero, il device
;verra' collocato nella lista dei device mantenuta
;dall'Exec
;--------------------------------------------------

initRoutine:          ; (d0:dev.ptr. A0:seg.list)
    ;mette il puntatore al device in A5
         move.l      a5,-(sp)
         move.l      d0,a5

    ;salva il puntatore a exec
         move.l      a6,md_ExecBase(a5)

    ;salva il puntatore alla segment list
         move.l      a0,md_SegList(a5)

    ;apre la dos.library
         lea         dosName(pc),a1
         CLEAR       d0
         CALLSYS     OpenLibrary

         move.l      d0,md_DosBase(a5)
         bne.s       init_DosOK

    ;non puo' aprire la dos.library
         ALERT       AG_OpenLib!AO_DOSLib

init_DosOK:
    ;genera i dati statici di cui ha bisogno
    ;
    ;le inizializzazioni specifiche del device devono
    ;essere poste qui.
    ;

         move.l      a5,d0
         move.l      (sp)+,a5
```

```
        rts

;--------------------------------------------------
;I comandi di interfacciamento del sistema iniziano
;qui. Le chiamate a OpenDevice()/CloseDevice/
;RemDevice() vengono tradotte in chiamate alle
;routine Open/Close/Expunge. Il puntatore alla base
;della libreria del device si trova in A6. Il task
;switching e' stato disabilitato con Forbid/Permit.
;--------------------------------------------------


;--------------------------------------------------
;Se c'e' un errore il campo IO_ERROR della struttura
;device request e' impostato, altrimenti nel campo
;IO_UNIT si trova un puntatore alla device unit.
;--------------------------------------------------

Open:                   ;device: A6
                        ;iob: A1
                        ;unitnum: D0;
                        ;flags: D1
        movem.l   d2/a2-a4,-(sp)

        move.l    a1,a2             ;salva l'iob

    ;lo unit number e' nell'intervallo
        moveq     #MD_NUMUNITS,d2
        cmp.l     d2,d0
        bcc.s     OpenError         ;unit number
                                    ;oltre
                                    ;l'intervallo

    ;la unit e' gia' inizializzata
        move.l    d0,d2             ;salva lo unit
                                    ;number
        lsl.l     #2,d0
        lea.l     md_Units(a6,d0.l),a4
        move.l    (a4),d0
        bne.s     OpenUnitOK

    ;prova a ottenere una unit
        bsr       InitUnit

    ;l'inizializzazione e' OK
        move.l    (a4),d0
        beq.s     OpenError

OpenUnitOK:
        move.l    d0,a3            ;puntatore alla
                                   ;unit in a3

        move.l    d0,IO_UNIT(a2)

    ;incrememta il contatore di unit aperte
        addq.w    #1,LIB_OPENCNT(a6)
        addq.w    #1,UNIT_OPENCNT(a3)

    ;nessun expunge ritardato permesso
        bclr      #LIBB_DELEXP,md_Flags(a6)

OpenEnd:
        movem.l   (sp)+,d2/a2-a4
        rts

OpenError:
        move.b    #IOERR_OPENFAIL,IO_ERROR(a2)
        bra.s     OpenEnd


;--------------------------------------------------
;Se il device non e' piu' aperto, Close restituisce
;la segment list se il flag di expunge ritardato e'
;impostato; altrimenti restituisce NULL.
;--------------------------------------------------


Close:    ;(device: A6, iob: A1)
        movem.l   a2-a3,-(sp)

        move.l    a1,a2

        move.l    IO_UNIT(a2),a3

    ;l'iob non puo' essere usato nuovamente
        moveq.l   #-1,d0
        move.l    d0,IO_UNIT(a2)
        move.l    d0,IO_DEVICE(a2)

    ;la unit e' ancora in uso
        subq.w    #1,UNIT_OPENCNT(a3)
        bne.s     CloseDevice

        bsr       ExpungeUnit

CloseDevice:
    ;decrementa il contatore delle aperture del
    ;device
        subq.w    #1,LIB_OPENCNT(a6)

    ;qualcuno ha il device aperto
        bne.s     CloseEnd

    ;c'e' qualche expunge ritardato
        btst      #LIBB_DELEXP,md_Flags(a6)
        beq.s     CloseEnd

    ;esegue l'expunge
        bsr       Expunge

CloseEnd:
        movem.l   (sp)+,a2-a3
        rts


;--------------------------------------------------
;Se il device non e' piu' aperto, Expunge ritorna la
;segment list del codice del device. Altrimenti
;viene impostato il flag di expunge ritardato e
```

```
;viene restituito NULL in D0.
;--------------------------------------------------

Expunge:      ;(device: A6)
        movem.l     d2/a5-a6,-(sp)
        move.l      a6,a5
        move.l      md_ExecBase(a5),a6

    ;il device e' aperto
        tst.w       LIB_OPENCNT(a5)
        beq         1$

    ;ancora aperto - imposta il flag di expunge
    ;ritardato
        bset        #LIBB_DELEXP,md_Flags(a5)
        CLEAR       d0
        bra.s       ExpungeEnd

1$:
    ;OK, rimuove il device - memorizza seglist in d2
        move.l      md_SegList(a5),d2

    ;unlink dalla lista dei device
        move.l      a5,a1
        CALLSYS     Remove

    ;
    ;Chiusure specifiche del device devono essere
    ;poste qui
    ;

    ;chiude la dos library
        move.l      md_DosBase(a5),a1
        CALLSYS     CloseLibrary

    ;libera la memoria allocata
        CLEAR       d0
        move.l      a5,a1
        move.w      LIB_NEGSIZE(a5),d0

        sub.l       d0,a1
        add.w       LIB_POSSIZE(a5),d0

        CALLSYS     FreeMem

    ;imposta il valore di ritorno
        move.l      d2,d0

ExpungeEnd:
        movem.l     (sp)+,d2/a5-a6
        rts


;--------------------------------------------------
;ExtFunc mette solamente 0 nel registro D0
;--------------------------------------------------


ExtFunc:
        CLEAR       d0
        rts


;--------------------------------------------------
;InitUnit inizializza una device unit e fa partire
;lo unit task associato.
;--------------------------------------------------

InitUnit:                   ; unit number: D2
                            ; scratch: A3
                            ; devptr: A6)
        movem.l     d2-d4,-(sp)

    ;alloca la memoria per la unit
        move.l      #MyDevUnit_Sizeof,d0
        move.l      #MEMF_PUBLIC!MEMF_CLEAR,d1
        LINKSYS     AllocMem,md_ExecBase(a6)

        tst.l       d0
        beq         InitUnitEnd

        move.l      d0,a3
        move.b      d2,mdu_UnitNum(a3) ;inizializza
                                       ;lo unit
                                       ;number
        move.b      #NT_MSGPORT,LN_TYPE(a3)

    ;Il processo unit viene iniziato qui. Si imposta
    ;il flag della message port del processo a
    ;PA_IGNORE finche' il flag non viene resettato
    ;dal nuovo processo.
        move.l      #MYPROCSTACKSIZE,d4    ;imposta
                                       ;la dimensione
                                       ;dello stack
        move.l      #myproc_seglist,d3     ;imposta
                                       ;la segment
                                       ;list
        lsr.l       #2,d3              ;cambia il
                                       ;puntatore
                                       ;BCPL
        moveq       #MYPROCPRI,d2      ;imposta la
                                       ;priorita' del
                                       ;processo
        move.l      #myName,d1
        LINKSYS     CreateProc,md_DosBase(a6)
        tst.l       d0
        beq         InitUnitFreeUnit

    ;imposta le strutture unit per il nuovo processo
        move.l      d0,a0
        move.l      d0,mdu_Process(a3)
        lea         -pr_MsgPort(a0),a0
        move.b      #PA_IGNORE,MP_FLAGS(a3)
        move.l      a0,MP_SIGTASK(a3)

    ;manda il messaggio di inizio al nuovo processo
        lea         mdu_Msg(a3),a1
        move.l      a3,mdm_Unit(a1)
        move.l      a6,mdm_Device(a1)
        move.b      #NT_MESSAGE,LN_TYPE(a1)
```

```
        move.l      d0,a0
        LINKSYS     PutMsg,md_ExecBase(a6)

    ;marca la unit come pronta per l'esecuzione
        move.b      mdu_UnitNum(a3),d0
        lsl.l       #2,d0
        move.l      a3,md_Units(a6,d0.l)

InitUnitEnd:
        movem.l     (sp)+,d2-d4
        rts

    ;errore - libera la struttura unit allocata in
    ;precedenza
InitUnitFreeUnit:
        bsr         FreeUnit
        bra.s       InitUnitEnd

    ;libera la memoria allocata per la unit
FreeUnit:           ;(unitptr: A3, deviceptr: A6)
        move.l      a3,a1
        move.l      #MyDevUnit_Sizeof,d0
        LINKSYS     FreeMem,md_ExecBase(a6)
        rts


ExpungeUnit:        ;(unitptr:A3, deviceptr:A6)
        move.l      d2,-(sp)

    ;rimuove il task della unit - a questo punto il
    ;contatore delle unita' aperte vale zero. Quindi
    ;e' abbastanza sicuro.
        move.l      mdu_Process(a3),a1
        lea         -(pr_MsgPort)(a1),a1
        LINKSYS     RemTask,md_ExecBase(a6)

    ;salva lo unit number
        CLEAR       d2
        move.b      mdu_UnitNum(a3),d2

    ;libera la struttura unit
        bsr         FreeUnit

    ;pulisce lo unit vector nella struttura del
    ;device
        lsl.l       #2,d2
        clr.l       md_Units(a6,d2.l)

        move.l      (sp)+,d2

        rts


;----------------------------------------------------
;I comandi specifici del device cominciano qui
;----------------------------------------------------


;----------------------------------------------------
;smdTable - qui vengono mantenuti gli indirizzi
;delle routine che implementano i comandi
;Gli indirizzi sono acquisiti con la chiamata
;ExecuteIO
;----------------------------------------------------

cmdTable:
        dc.l    Invalid         ;$00000001  bit 0
        dc.l    Reset           ;$00000002      1
        dc.l    Read            ;$00000004      2
        dc.l    Write           ;$00000008      3
        dc.l    Update          ;$00000010      4
        dc.l    Clear           ;$00000020      5
        dc.l    Stop            ;$00000040      6
        dc.l    Start           ;$00000080      7
        dc.l    Flush           ;$00000100      8
        dc.l    Custom          ;$00000200      9

;Quali comandi non devono essere accodati
IMMEDIATE   EQU     $000001c2       ;Flush|Start|
                                    ;MyStop|MyReset


;----------------------------------------------------
;BeginIO - tutto l'IO arriva attravero questa
;routine. L'IO viene accodato o eseguito
;immediatamente
;----------------------------------------------------

BeginIO:            ;(iob:A1, device:A6)
        move.l      a3,-(sp)

    ;copia la device unit
        move.l      IO_UNIT(a1),a3

    ;Il comando di IO e' previsto
        CLEAR       d0
        move.w      IO_COMMAND(a1),d0
        cmp.w       #MYDEV_END,d0
        bcc.s       BeginIO_BadCmd

        DISABLE     a0

    ;gestisce i comandi con il quick flag impostato
        btst        #IOB_QUICK,IO_FLAGS(a1)
        bne.s       BeginIO_Immediate

    ;gestisce tutti i comandi immediati
        move.w      #IMMEDIATE,d1
        btst        d0,d1
        bne.s       BeginIO_Immediate

    ;l'unita' e' ferma - se e' cosi' accoda il
    ;messaggio
        btst        #MDUB_STOPPED,UNIT_FLAGS(a3)
        bne.s       BeginIO_QueueMsg

    ;non e' un comando immediato - il device e'
    ;occupato
        bset        #UNITB_ACTIVE,UNIT_FLAGS(a3)
        beq.s       BeginIO_Immediate
```

```
    ;accoda la richiesta - marca il device come
    ;avente bisogno dell'attenzione del task
    ;pulisce il quick flag

BeginIO_QueueMsg:
        bset        #UNITB_INTASK,UNIT_FLAGS(a3)
        bclr        #IOB_QUICK,IO_FLAGS(a1)

        ENABLE      a0

        move.l      mdu_Msg(a3),a2
        move.l      a1,mdm_Iob(a2)
        move.l      a2,a1
        move.l      mdu_Process(a3),a0
        LINKSYS     PutMsg,md_ExecBase(a6)
        bra.s       BeginIO_End

BeginIO_Immediate:
        ENABLE      a0
        bsr         ExecuteIO

BeginIO_End:
        move.l      (sp)+,a3
        rts

BeginIO_BadCmd:
        move.b      #IOERR_NOCMD,IO_ERROR(a1)
        bra.s       BeginIO_End


;---------------------------------------------------
;ExecuteIO fa l'effettivo dispatching delle
;richieste. A3 contiene il puntatore alla unit, A6
;quello al device e A1 ha la richiesta di IO.
;---------------------------------------------------

ExecuteIO:      ;(iob:A1, unitptr:A3, devptr:A6)
        move.l      a2,-(sp)
        move.l      a1,a2

        move.b      #0,IO_ERROR(a2)      ;pulisce il
                                         ;campo dell'errore

        CLEAR       d0
        move.w      IO_COMMAND(a2),d0
        lsl.l       #2,d0               ;byte offset
                                        ;nella tabella
        lea         cmdTable(pc),a0
        move.l      0(a0,d0.w),a0

        jsr         (a0)

        move.l      (sp)+,a2
        rts


;---------------------------------------------------
;TerminateIO manda indietro la richiesta al
;chiamante. Il device non e' impostato come inattivo
;se la richiesta era immediata o se era stata fatta
;partire dal task della unit.
;---------------------------------------------------

TerminateIO:        ;(iob:A1, unitptr:A3, devptr:A6)
        CLEAR       d0
        move.w      IO_COMMAND(a1),d0
        move.w      #IMMEDIATE,d1
        btst        d0,d1
        bne.s       TerminateIO_Immediate

    ;spegne il bit active?
        btst        #UNITB_INTASK,UNIT_FLAGS(a3)
        bne.s       TerminateIO_Immediate

    ;non c'e' altro lavoro per il task
        bclr        #UNITB_ACTIVE,UNIT_FLAGS(a3)

TerminateIO_Immediate:
    ;Se il bit quick io e' impostato ritorna al
    ;chiamante senza un messaggio di risposta
        btst        #IOB_QUICK,IO_FLAGS(a1)
        bne.s       TerminateIO_End

        LINKSYS     ReplyMsg,md_ExecBase(a6)

TerminateIO_End:
        rts


;---------------------------------------------------
;AbortIO - Non e' implementata
;---------------------------------------------------

AbortIO:            ;(iob:A1, device:A6)
        rts

AbortIO_End:
        rts


;---------------------------------------------------
;Le funzioni che implementano i comandi del device
;iniziano qui. Viene adottato il seguente formato
;per i registri.
;   A1 - ptr. all'io request block
;   A2 - un altro ptr.all'iob
;   A3 - ptr. alla struttura unit
;   A6 - ptr. alla struttura device
;---------------------------------------------------


;---------------------------------------------------
;Invalid - ritorna semplicemente IOERR_NOCMD nel
;campo io_error
;---------------------------------------------------

Invalid:
        move.b      #IOERR_NOCMD,IO_ERROR(a1)
        move.l      #0,IO_ACTUAL(a1)
```

```
        bsr         TerminateIO
        rts


;-------------------------------------------------
;Reset - Non e' implementato
;-------------------------------------------------

Reset:
        move.l      #0,IO_ACTUAL(a1)
        rts


;-------------------------------------------------
;Read - agisce come un infinita sorgente di 1.
;Restituisce il numero 1 nel buffer
;-------------------------------------------------

Read:
        move.l      IO_DATA(a1),a0
        move.l      IO_LENGTH(a1),d0
        CLEAR       d2
        move        #0,ccr

    ;gestisce read con lunghezza zero
        beq.s       ReadEnd

    ;copia il dato
        CLEAR       d1
    ;valore in D1 per la copia
        move.l  #1,d1

ReadLoop:
        move.b      d1,(a0)+
        add.l       #1,d2
        sub.l       #1,d0
        bne.s       ReadLoop        ;se Z non e'
                                    ;settato (d0!=0)

ReadEnd:
        move.l      d2,IO_ACTUAL(a1)
        bsr         TerminateIO
        rts


;-------------------------------------------------
;Write - fa un dump del buffer della richiesta.
;Restituisce il numero dei byte scritti nel campo
;io_actual
;-------------------------------------------------

Write:
        move.l      IO_LENGTH(a1),IO_ACTUAL(a1)
        bsr         TerminateIO
        rts


;-------------------------------------------------
;Update - non implementato - dovrebbe scrivere tutto
;l'IO in attesa, senza ritornare finche' non ha
;finito
;-------------------------------------------------


Update:
        bra         Invalid


;-------------------------------------------------
;Clear - non implementato - dovrebbe pulire tutti i
;buffer interni
;-------------------------------------------------

Clear:
        bra         Invalid


;-------------------------------------------------
;Stop - ferma il trattamento di tutte le richieste
;di IO finche' non viene ricevuto il comando Start
;-------------------------------------------------

Stop:
        bset        #MDUB_STOPPED,UNIT_FLAGS(a3)
        move.l      #0,IO_ACTUAL(a1)
        bsr         TerminateIO
        rts


;-------------------------------------------------
;Start - fa ripartire un task di una unit
;precedentemente fermato
;-------------------------------------------------

Start:
        bsr         InternalStart
        move.l      a2,a1
        move.l      #0,IO_ACTUAL(a1)
        bsr         TerminateIO
        rts

InternalStart:
    ;riattiva il trattamento delle richieste
        bclr        #MDUB_STOPPED,UNIT_FLAGS(a3)

    ;segnala al task di ripartire
        move.l      a3,a1
        CLEAR       d0

        CLEAR       d1
        move.b      MP_SIGBIT(a3),d1
        bset        d1,d0
        LINKSYS     Signal,md_ExecBase(a6)

        rts


;-------------------------------------------------
;Flush - restituisce tutte le richieste in attesa ai
```

```
;rispettivi chiamanti.
;----------------------------------------------------

Flush:
        movem.l     d2/a6,-(sp)

        move.l      md_ExecBase(a6),a6

        bset        #MDUB_STOPPED,UNIT_FLAGS(a3)
        sne         d2

FlushLoop:
    ;va in loop finche' non ci sono piu' richieste
        move.l      a3,a0
        CALLSYS     GetMsg

        tst.l       d0
        beq.s       FlushEnd

        move.l      d0,a1
        move.b      #IOERR_ABORTED,IO_ERROR(a1)
        CALLSYS     ReplyMsg

        bra.s       FlushLoop

FlushEnd:
        move.l      d2,d0
        movem.l     (sp)+,d2/a6
        tst.b       d0
        beq.s       1$

        bsr         InternalStart

1$:
        move.l      a2,a1
        bsr         TerminateIO

        rts


;----------------------------------------------------
;Custom - comando custom - non fa molto
;----------------------------------------------------

Custom:
        bsr     TerminateIO
        rts


;----------------------------------------------------
;Le funzioni del device iniziano qui
;----------------------------------------------------


;----------------------------------------------------
;DevFunc - restituisce il contatore delle aperture
;per una unit specifca
;----------------------------------------------------
DevFunc:      ;(devptr:A6, unit no.:D0)
        move.l      a3,-(sp)

    ;unit number nell'intervallo
        move.l      #MD_NUMUNITS,d2
        cmp.l       d2,d0
        bcc.s       DevFuncError

        lsl.l       #2,d0
        move.l      md_Units(a6,d0.l),a3
        cmp.l       #0,a3
        beq.s       DevFuncError

        move.w      UNIT_OPENCNT(a3),d0

DevFuncExit:
        move.l      (sp)+,a3
        rts

DevFuncError:
        move.l      #-1,d0                  ;flag
error
        bra.s       DevFuncExit


;----------------------------------------------------
;Il processo della unit del device inizia qui
;
;Viene usato un processo in modo che le richieste
;accodate possano essere trattate in un momento
;successivo.
;----------------------------------------------------

;----------------------------------------------------
;Questo pezzo imbroglia il DOS - abbiamo alterato il
;codice in memoria usando LoadSeg. Usando Loadseg il
;segmento ha il seguente prefisso:
;  <seg.len><ptr.to next seg.>
;Abbiamo impostato la segment list qui, e il
processo
;inizia al primo indirizzo dopo la segment list
;----------------------------------------------------

MyDevProcStart:
 CNOP     0,4               ;allineamento su long
word
        dc.l        16      ;lunghezza del segmento
MyDevProcSeglist:
        dc.l        0       ;ptr. al prossimo
segmento


;----------------------------------------------------
;Il processo inizia qui
;----------------------------------------------------

MyDevProcBegin:
        move.l      _AbsExecBase,a6
```

```
    ;aspetta il messaggio di startup
        sub.l       a1,a1
        CALLSYS     FindTask
        move.l      d0,a0
        move.l      d0,a4
        lea         pr_MsgPort(a0),a0
        move.l      a0,d4
        CALLSYS     WaitPort

    ;prende il messaggio dalla msg port - msgport e'
    ;marcata con PA_IGNORE
        move.l      d0,a1
        move.l      d0,d2
        CALLSYS     Remove

    ;prende i parametri necessari
        move.l      d2,a2
        move.l      mdm_Device(a2),a5 ;a5 e' il
                                      ;nostro nuovo
                                      ;device
        move.l      mdm_Unit(a2),a3

        move.l      a5,d5       ;salva il device ptr.
        move.l      a3,d3       ;salva lo unit ptr.

    ;alloca un segnale
        moveq       #-1,d0       ;-1 e': nessun
                                 ;segnale del tutto
        CALLSYS     AllocSignal
        move.l      d3,a3        ;recupera lo unit
                                 ;ptr.
        move.b      d0,MP_SIGBIT(a3)
        move.b      #PA_SIGNAL,MP_FLAGS(a3)

;----------------------------------------------------
;Il loop principale inizia qui.
;Procedura:
;  - aspetta un messaggio alla porta
;  - fa un lock del device - il task e' attivo e sta
;    trattando le richieste
;  - prende il messaggio - se non c'e' messaggio
;    fa un unlock del device
;  - chiama ExecuteIO e tratta il messaggio
;  - riprende da capo aspettando un messaggio
;----------------------------------------------------


        bra.s       MyDevProcCheckStatus

MyDevProcMainLoop:
    ;aspetta un messaggio alla porta
        move.l      d4,a0           ;MsgPort per
                                    ;questo processo
        CALLSYS     WaitPort

MyDevProcCheckStatus:
    ;il flag stopped del processo e' settato ?
        move.l      d3,a3
        btst        #MDUB_STOPPED,UNIT_FLAGS(a3)
        bne.s       MyDevProcMainLoop

    ;setta il flag active del processo - quindi
    ;vengono accodati i messaggi
        bset        #UNITB_ACTIVE,UNIT_FLAGS(a3)
        beq.s       MyDevProcMainLoop

MyDevProcNextMessage:
    ;prende la prossima richiesta
        move.l      d4,a0
        CALLSYS     GetMsg
        tst.l       d0
        beq.s       MyDevProcUnlock     ;messaggio?

    ;handle the request
        move.l      d0,a2
        move.l      d3,a3           ;MyDevUnit ptr.
        move.l      mdm_Iob(a2),a1  ;iob ptr.
        exg         a5,a6
        bsr         ExecuteIO
        exg         a5,a6

        bra.s       MyDevProcNextMessage

   ;non ci sono altri messaggi - possiamo aspettare
   ;per altri
MyDevProcUnlock:
        and.b     #$ff&(UNITB_ACTIVE!UNITB_INTASK),
                  UNIT_FLAGS(a3)
        bra         MyDevProcMainLoop

   ;arriviamo qui se fallisce l'inizializzazione
MyDevProcFail:
        bsr         FreeUnit
        rts


;----------------------------------------------------
;Questa e' la fine del codice del nostro device
;----------------------------------------------------

EndDevCode:
        END
```

Listato 4: add_dev.c

```
/***************************************************
 * add_dev.c
 *
 * Modulo utilizzato per aggiungere un device
 * dell'utente alla lista dei device mantenuta dal
 * sistema. Dopo aver aggiunto il device dell'utente
 * e' disponibile a tutti i programmi seguenti.
 *
 * Storia:
```

```
 *   88-05-06  creato
 *
 * compilazione:  cc add_dev
 * link:          ln add_dev -lc
 ************************************************/


#include <functions.h>
#include <exec/types.h>
#include <exec/libraries.h>
#include <exec/resident.h>
#include <exec/devices.h>
#include <libraries/dos.h>
#include <libraries/dosextens.h>


ULONG segList,      /* puntatore alla seglist BCPL */
      codeloc;      /* puntatore alla locazione del
                       codice */

struct Init {
  ULONG space;        /* dimensione dello spazio
dati
                         */
  ULONG funcTable;    /* ptr. alla tabella degli
                          offset delle funzioni */
  ULONG dataTable;    /* ptr. alla tabella
initStruct
                         */
  ULONG initRoutine;  /* ptr. alla library
                          initRoutine */
} *init;

struct Resident *devRes;  /* ptr. all'area rom tag
                             del device */

ULONG space, funcTable, dataTable, initRoutine;

struct Device *mydev;


main()
{

    /* carica il segmento del device */
  segList = (ULONG)LoadSeg("mydev.device");
  if (segList==NULL) {
    printf("Non posso caricare mydev.device\n");
    exit(0);
  }

    /* indirizzo dell'inizio del codice del device
     * in memoria, moltiplicate per 4 perche'
     * segList e' un puntatore BCPL */

  codeloc = segList * 4;

    /* indirizzo dell'area residente rom tag */
  devRes = (struct Resident *)(codeloc + 8);

    /* indirizzo della tabella di inizializzazione */
  init = (struct Init *)devRes->rt_Init;

    /* estrae i dati dalla tabella di
     * inizializzazione */
  space = init->space;
  funcTable = init->funcTable;
  dataTable = init->dataTable;
  initRoutine = init->initRoutine;

    /* inizializza il device */
  mydev = (struct Device *)MakeLibrary(funcTable,
           dataTable, initRoutine, space, segList);

     /* rende il device conosciuto al sistema */
  AddDevice(mydev);

  printf("mydev.device aggiunto al sistema\n");

}
```

Listato 5: main.c

```
/************************************************
 * main.c
 *
 * Programma di test per mydev.device
 *
 * Storia:
 *   88-09-11  creato
 *
 * compilazione:  cc main
 * link:          ln main devfunc -lc
 ************************************************/


#include <functions.h>
#include <exec/types.h>
#include <exec/libraries.h>
#include <exec/devices.h>
#include <exec/io.h>

#define CMD_CUSTOM  (CMD_FLUSH+1)


struct MsgPort *reply;
struct IOStdReq *req;
UBYTE dev_flag=0;
UBYTE buff[10];

main()
{
  ULONG ret, unit, val, no_open, DevFunc();
  WORD j;

    /* crea la reply port */
```

```
  reply = CreatePort("my_main_reply",0);
  if (reply==NULL) {
    printf("Non posso creare la reply port\n");
    cleanup();
  }

    /* crea la io request */
  req = (struct IOStdReq *)CreateStdIO(reply);
  if (req==NULL) {
    printf("Non posso creare la IO request\n");
    cleanup();
  }

    /* apre il device di prova */
  ret = OpenDevice("mydev.device",0L,req,0L);
  if (ret!=NULL) {
    printf("Non posso aprire mydev.device\n");
    cleanup();
  }
  dev_flag = 1;

    /* legge qualcosa */
  req->io_Command = CMD_READ;
  req->io_Data = (APTR)buff;
  req->io_Length = (ULONG)sizeof(buff);
  DoIO(req);
  printf("io_Actual=%ld\n",req->io_Actual);

    /* ora prova i nostri comandi custom */
  req->io_Command = CMD_CUSTOM;
  BeginIO(req);
  printf("io_Actual=%ld\n",req->io_Actual);

    /* chiama la nostra funzione custom */
  unit = 0L;
  no_open = DevFunc(req,unit);
  printf("no_open=%ld\n",no_open);

  cleanup();

}


/***************************************************
 * cleanup() - rilascia ogni cosa che sia stata
 *             aperta o allocata
 ***************************************************/


cleanup()
{

  if (dev_flag) {
    CloseDevice(req);
  }
  if (req) {
    DeleteStdIO(req);
  }
  if (reply) {
    DeletePort(reply);
    }

}
```

Listato 6: devfunc.asm

```
;-------------------------------------------------
; devfunc.asm
;
; routine di interfacciamento tra il programma C e
; la funzione in mydev.device
;
; Storia:
;   88-09-11  creato
;
; assemblaggio:  as devfunc
;-------------------------------------------------

 NOLIST
 INCLUDE "exec/types.i"
 INCLUDE "exec/libraries.i"
 INCLUDE "exec/nodes.i"
 INCLUDE "exec/lists.i"
 INCLUDE "exec/io.i"
 INCLUDE "exec/devices.i"
 INCLUDE "asmsupp.i"
 INCLUDE "mydev_def.i"

 LIST

 XREF  _DevFunc

;-------------------------------------------------
; Sequenza di chiamata C:
;                      no_open = DevFunc(req,unit)
; registri:              D0             A0   D0
;
;                      LONG no_open, unit;
;                      struct IOStdReq *req;
;
;-------------------------------------------------

_DevFunc:
    ;recupera i dati della chiamata dallo stack
         move.l      4(sp),a0
         move.l      8(sp),d0

    ;chiama la funzione del device
         LINKLIB  Dev_Func,IO_DEVICE(a0)

    ;ritorna al chiamante
         rts

         end
```

CEDOLE SERVIZIO LETTORI

ABBONAMENTO RIVISTE JACKSON

ATTENZIONE Questa cartolina riporta sul retro un modulo speciale con una serie di domande a cui preghiamo vivamente di rispondere con precisione.

ORDINI QUI SOTTO CIÒ CHE HA SCELTO

AVVERTENZA I rinnovi entreranno in vigore automaticamente a partire dal numero successivo alla data di scadenza dell'abbonamento precedente.
Per motivi tecnici, i nuovi abbonamenti saranno attivati dopo 6 settimane dalla data di ricevimento della cartolina di richiesta, o in data successiva nel caso di esplicita comunicazione da parte dell'abbonato.

N.B. Per abbonamenti all'estero le tariffe dovranno essere raddoppiate. Non è prevista la spedizione via aerea

☐ **NUOVO ABBONAMENTO** ☐ **RINNOVO**

Rivista	Numeri	Prezzo
☐ EO NEWS SETTIMANALE	numeri 40 + 6 omaggio	L. 59.500
☐ ELETTRONICA OGGI	numeri 20	L. 80.500
☐ AUTOMAZIONE OGGI	numeri 20	L. 80.000
☐ MECCANICA OGGI	numeri 11	L. 36.000
☐ STRUMENTAZIONE E MISURE OGGI	numeri 11	L. 52.000
☐ INFORMATICA OGGI SETTIMANALE	numeri 40 + 6 omaggio	L. 61.000
☐ INFORMATICA OGGI MESE	numeri 11	L. 44.500
☐ BIT	numeri 11	L. 48.000
☐ PC MAGAZINE	numeri 11	L. 43.500
☐ PC FLOPPY	numeri 11	L. 106.000
☐ COMPUTERGRAFICA & DESKTOP PUBLISHING	numeri 11	L. 52.500
☐ TRASMISSIONE DATI E TELECOM.	numeri 11	L. 45.500
☐ NTE COMPUSCUOLA	numeri 10	L. 32.500
☐ WATT	numeri 20	L. 36.500
☐ MEDIA PRODUCTION	numeri 11	L. 62.000
☐ STRUMENTI MUSICALI	numeri 11	L. 44.000
☐ FARE ELETTRONICA	numeri 12	L. 58.000
☐ AMIGA MAGAZINE	numeri 11	L. 123.500
☐ AMIGA TRANSACTOR	numeri 6	L. 34.000
☐ SUPERCOMMODORE 64/128 disk	numeri 11	L. 105.500
☐ SUPERCOMMODORE 64/128 tape	numeri 11	L. 66.000
☐ OLIVETTI PRODEST USER	numeri 6	L. 24.000
☐ PC SOFTWARE	numeri 11	L. 88.000
☐ PC GAMES 5¼"	numeri 11	L. 124.000
☐ PC GAMES 3½"	numeri 11	L. 132.500
☐ 3½" SOFTWARE	numeri 11	L. 132.000

MODALITÀ DI PAGAMENTO

☐ Allego assegno n. ______ di L. ______
Banca ______
☐ Ho effettuato versamento di L. ______ sul c/c postale n. 11666203 intestato a Gruppo Editoriale Jackson - Milano e allego fotocopia della ricevuta.
☐ Ho effettuato versamento di L. ______ tramite vaglia postale o telegrafico e allego fotocopia della ricevuta.
☐ Vi autorizzo ad addebitare l'importo di L. ______ sulla carta di credito ☐ VISA ☐ AMERICAN EXPRESS ☐ DINERS CLUB ☐ CARTA SI
N. ______ Data di scadenza a ______
Data ______ Firma ______

COMMISSIONE LIBRI

ATTENZIONE Questa cartolina riporta sul retro un modulo speciale con una serie di domande a cui preghiamo vivamente di rispondere con precisione.

ORDINI QUI SOTTO CIÒ CHE HA SCELTO

Codice articolo	Titolo abbreviato	Prezzo L.
	Contributo fisso per spese di spedizione	L. 4.000
	TOTALE	

Ordine minimo £ 50.000 + £ 4.000 per contributo fisso spese di spedizione
☐ **Non sono abbonato** a riviste Jackson
☐ **Sono abbonato** alla/e seguente/i rivista/e Jackson ______ e ho quindi diritto allo sconto del 20% sino al 28/2/89 e del 10% dopo tale data

MODALITÀ DI PAGAMENTO

☐ Contro assegno postale ☐ Assegno allegato n° ______
Banca ______
☐ Versamento su c/c postale 11666203 a Voi intestato del quale allego fotocopia della ricevuta
☐ Versamento con vaglia postale a Voi intestato del quale allego fotocopia della ricevuta
☐ Carta di Credito ☐ American Express ☐ Visa ☐ Diners Club
☐ Carta Sì Autorizzo l'organizzazione sopra indicata ad addebitare l'importo sul mio Conto n° ______ Scadenza ______
☐ Richiedo l'emissione della fattura (formula riservata alle aziende) e comunico il numero di Partita I.V.A.
Data ______ Firma ______

ARRETRATI RIVISTE

ATTENZIONE Questa cartolina riporta sul retro un modulo speciale con una serie di domande a cui preghiamo vivamente di rispondere con precisione.

ORDINI QUI SOTTO CIÒ CHE HA SCELTO

Rivista	Numero	Anno	Prezzo
☐ INDUSTRIA OGGI	Numero ______	Anno ______	L. 10.000 cad.
☐ ELETTRONICA OGGI	Numero ______	Anno ______	L. 10.000 cad.
☐ AUTOMAZIONE OGGI	Numero ______	Anno ______	L. 10.000 cad.
☐ EO NEWS SETTIMANALE	Numero ______	Anno ______	L. 4.000 cad.
☐ WATT	Numero ______	Anno ______	L. 4.000 cad.
☐ TRASMISSIONE DATI E TELECOMUNICAZIONI	Numero ______	Anno ______	L. 10.000 cad.
☐ VIDEOTEL MAGAZINE	Numero ______	Anno ______	L. 8.000 cad.
☐ INFORMATICA OGGI	Numero ______	Anno ______	L. 10.000 cad.
☐ INFORMATICA OGGI SETTIMANALE	Numero ______	Anno ______	L. 4.000 cad.
☐ LAB NEWS	Numero ______	Anno ______	L. 10.000 cad.
☐ COMPUTER GRAFICA E APPLICAZIONI	Numero ______	Anno ______	L. 12.000 cad.
☐ PC MAGAZINE	Numero ______	Anno ______	L. 10.000 cad.
☐ PC MAGAZINE + PC FLOPPY	Numero ______	Anno ______	L. 24.000 cad.
☐ PC GAMES 5 ¼	Numero ______	Anno ______	L. 24.000 cad.
☐ PC GAMES 3½	Numero ______	Anno ______	L. 30.000 cad.
☐ PC 3½ SOFTWARE	Numero ______	Anno ______	L. 30.000 cad.
☐ BIT	Numero ______	Anno ______	L. 10.000 cad.
☐ SUPERCOMMODORE 64/128 (cassetta)	Numero ______	Anno ______	L. 15.000 cad.
☐ SUPERCOMMODORE 64/128 (disco)	Numero ______	Anno ______	L. 24.000 cad.
☐ NOI 128/64 (cassetta)	Numero ______	Anno ______	L. 18.000 cad.
☐ NOI 128/64 (disco)	Numero ______	Anno ______	L. 26.000 cad.
☐ COMMODORE PROFESSIONAL (cassetta)	Numero ______	Anno ______	L. 18.000 cad.
☐ COMMODORE PROFESSIONAL (disco)	Numero ______	Anno ______	L. 26.000 cad.
☐ AMIGA MAGAZINE (disco)	Numero ______	Anno ______	L. 28.000 cad.
☐ AMIGA TRANSACTOR	Numero ______	Anno ______	L. 14.000 cad.
☐ OLIVETTI PRODEST USER	Numero ______	Anno ______	L. 10.000 cad.
☐ LA RIVISTA DI ATARI	Numero ______	Anno ______	L. 10.000 cad.
☐ COMPUSCUOLA	Numero ______	Anno ______	L. 8.000 cad.
☐ FARE ELETTRONICA	Numero ______	Anno ______	L. 10.000 cad.
☐ LABORATORIO DI ELETTRONICA PROFESSIONALE	Numero ______	Anno ______	L. 3.000 cad.
☐ STRUMENTI MUSICALI	Numero ______	Anno ______	L. 10.000 cad.
☐ NAUTICAL QUARTERLY	Numero ______	Anno ______	L. 20.000 cad.
☐ AUTOMOBILE QUARTERLY	Numero ______	Anno ______	L. 20.000 cad.

MODALITÀ DI PAGAMENTO

NB: Non si effettuano spedizioni contro assegno il pagamento deve sempre essere allegato all'ordine

☐ Allego assegno n° ______ di L. ______
Banca ______
☐ Allego l'importo in contanti di L. ______
☐ Versamento su c/c postale 11666203 a Voi intestato del quale allego fotocopia della ricevuta

COMPILARE E SPEDIRE IN BUSTA
CHIUSA A: GRUPPO EDITORIALE
JACKSON - VIA ROSELLINI 12 -
20124 MILANO

SERVIZIO QUALIFICAZIONE LETTORI

ATTENZIONE Questa cartolina riporta un modulo speciale con una serie di domande a cui preghiamo vivamente di rispondere con precisione.

INDIRIZZO PRIVATO

COGNOME E NOME ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ ANNO DI NASCITA 19____
TITOLO DI STUDIO: ☐ LAUREA ☐ MEDIA SUPERIORE ☐ MEDIA INFERIORE

INDIRIZZO LAVORO

DITTA O ENTE ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ TELEX ____________

ATTIVITÀ AZIENDA
A ☐ Informatica
B ☐ Automazione Industriale
C ☐ Meccanica
D ☐ Strumentazione elettronica
E ☐ Telecomunicazioni e Telefonia
F ☐ Elettronica
G ☐ Chimica
H ☐ Elettrotecnica e Impianti elettrici
I ☐ Laboratori di analisi Chimica e medica
L ☐ Altra industria manifatturiera
M ☐ Agricoltura
N ☐ Ingegneria/Edilizia/Architettura
O ☐ Finanza/Banche/Assicurazioni
P ☐ Editoria/Grafica/Pubblicità
Q ☐ Pubblica amministrazione centrale/Locale
R ☐ Consulenza legale/Commerciale
S ☐ Commercio/Distribuzione
T ☐ Istruzione (Scuola/Università)
U ☐ Formazione/Ricerca
V ☐ Broadcast/Audio e video professionale
Z ☐ Strumenti musicali
X ☐ Altro (specificare)

N. DI DIPENDENTI
A ☐ da 1 a 49 **C** ☐ da 250 a 999
B ☐ da 50 a 249 **D** ☐ da 1000 in su

FUNZIONI
AA ☐ Acquisti
BB ☐ Vendite
CC ☐ Progettazione/Ricerca e sviluppo
DD ☐ Marketing e Comunicazione

EE ☐ Direzione Generale
FF ☐ Produzione
GG ☐ Amministrazione/ Personale/Finanza
HH ☐ EDP
II ☐ Professionista/Consulente
LL ☐ Docente/Formatore
MM ☐ Studente
NN ☐ Altro (specificare)

INTERESSI PRINCIPALI
01 ☐ EDP
02 ☐ Personal Computer
03 ☐ Home Computer
04 ☐ Automazione Industriale e Meccanica
05 ☐ Strumentazione elettronica
06 ☐ Telecomunicazioni e telefonia
07 ☐ Elettronica professionale
08 ☐ Elettronica hobbystica
09 ☐ Elettrotecnica e impianti elettrici
10 ☐ Strumenti musicali
11 ☐ Marketing e management
12 ☐ Broadcast/audio e video professionale
13 ☐ Didattica
14 ☐ Altro (specificare)

CHE PERSONAL COMPUTER POSSIEDE
DOS ☐ MS DOS e compatibili
MAC ☐ Macintosh
AMG ☐ Amiga
C64 ☐ Commodore 64
VAR ☐ Altro home computer

COMPILARE E SPEDIRE IN BUSTA
CHIUSA A: GRUPPO EDITORIALE
JACKSON - VIA ROSELLINI 12 -
20124 MILANO

SERVIZIO QUALIFICAZIONE LETTORI

ABBONAMENTO GRATUITO
A 6 NUMERI, A SCELTA TRA LE SEGUENTI RIVISTE SETTIMANALI
☐ EO News Sett. ☐ INFORMATICA Oggi Sett.

BARRARE LA CASELLA RELATIVA ALLA RIVISTA PRESCELTA

INDIRIZZO PRIVATO

COGNOME E NOME ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ ANNO DI NASCITA 19____
TITOLO DI STUDIO: ☐ LAUREA ☐ MEDIA SUPERIORE ☐ MEDIA INFERIORE

INDIRIZZO LAVORO

DITTA O ENTE ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ TELEX ____________

ATTIVITÀ AZIENDA
A ☐ Informatica
B ☐ Automazione Industriale
C ☐ Meccanica
D ☐ Strumentazione elettronica
E ☐ Telecomunicazioni e Telefonia
F ☐ Elettronica
G ☐ Chimica
H ☐ Elettrotecnica e Impianti elettrici
I ☐ Laboratori di analisi Chimica e medica
L ☐ Altra industria manifatturiera
M ☐ Agricoltura
N ☐ Ingegneria/Edilizia/Architettura
O ☐ Finanza/Banche/Assicurazioni
P ☐ Editoria/Grafica/Pubblicità
Q ☐ Pubblica amministrazione centrale/Locale
R ☐ Consulenza legale/Commerciale
S ☐ Commercio/Distribuzione
T ☐ Istruzione (Scuola/Università)
U ☐ Formazione/Ricerca
V ☐ Broadcast/Audio e video professionale
Z ☐ Strumenti musicali
X ☐ Altro (specificare)

N. DI DIPENDENTI
A ☐ da 1 a 49 **C** ☐ da 250 a 999
B ☐ da 50 a 249 **D** ☐ da 1000 in su

FUNZIONI
AA ☐ Acquisti
BB ☐ Vendite
CC ☐ Progettazione/Ricerca e sviluppo
DD ☐ Marketing e Comunicazione

EE ☐ Direzione Generale
FF ☐ Produzione
GG ☐ Amministrazione/ Personale/Finanza
HH ☐ EDP
II ☐ Professionista/Consulente
LL ☐ Docente/Formatore
MM ☐ Studente
NN ☐ Altro (specificare)

INTERESSI PRINCIPALI
01 ☐ EDP
02 ☐ Personal Computer
03 ☐ Home Computer
04 ☐ Automazione Industriale e Meccanica
05 ☐ Strumentazione elettronica
06 ☐ Telecomunicazioni e telefonia
07 ☐ Elettronica professionale
08 ☐ Elettronica hobbystica
09 ☐ Elettrotecnica e impianti elettrici
10 ☐ Strumenti musicali
11 ☐ Marketing e management
12 ☐ Broadcast/audio e video professionale
13 ☐ Didattica
14 ☐ Altro (specificare)

CHE PERSONAL COMPUTER POSSIEDE
DOS ☐ MS DOS e compatibili
MAC ☐ Macintosh
AMG ☐ Amiga
C64 ☐ Commodore 64
VAR ☐ Altro home computer

COMPILARE E SPEDIRE IN BUSTA
CHIUSA A: GRUPPO EDITORIALE
JACKSON - VIA ROSELLINI 12 -
20124 MILANO

SERVIZIO QUALIFICAZIONE LETTORI

ABBONAMENTO GRATUITO
A 6 NUMERI, A SCELTA TRA LE SEGUENTI RIVISTE SETTIMANALI
☐ EO News Sett. ☐ INFORMATICA Oggi Sett.

BARRARE LA CASELLA RELATIVA ALLA RIVISTA PRESCELTA

INDIRIZZO PRIVATO

COGNOME E NOME ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ ANNO DI NASCITA 19____
TITOLO DI STUDIO: ☐ LAUREA ☐ MEDIA SUPERIORE ☐ MEDIA INFERIORE

INDIRIZZO LAVORO

DITTA O ENTE ____________
VIA E NUMERO ____________
CAP ______ CITTÀ ____________ PROV. ______
TEL. (______) ____________ TELEX ____________

ATTIVITÀ AZIENDA
A ☐ Informatica
B ☐ Automazione Industriale
C ☐ Meccanica
D ☐ Strumentazione elettronica
E ☐ Telecomunicazioni e Telefonia
F ☐ Elettronica
G ☐ Chimica
H ☐ Elettrotecnica e Impianti elettrici
I ☐ Laboratori di analisi Chimica e medica
L ☐ Altra industria manifatturiera
M ☐ Agricoltura
N ☐ Ingegneria/Edilizia/Architettura
O ☐ Finanza/Banche/Assicurazioni
P ☐ Editoria/Grafica/Pubblicità
Q ☐ Pubblica amministrazione centrale/Locale
R ☐ Consulenza legale/Commerciale
S ☐ Commercio/Distribuzione
T ☐ Istruzione (Scuola/Università)
U ☐ Formazione/Ricerca
V ☐ Broadcast/Audio e video professionale
Z ☐ Strumenti musicali
X ☐ Altro (specificare)

N. DI DIPENDENTI
A ☐ da 1 a 49 C ☐ da 250 a 999
B ☐ da 50 a 249 D ☐ da 1000 in su

FUNZIONI
AA ☐ Acquisti
BB ☐ Vendite
CC ☐ Progettazione/Ricerca e sviluppo
DD ☐ Marketing e Comunicazione

EE ☐ Direzione Generale
FF ☐ Produzione
GG ☐ Amministrazione/ Personale/Finanza
HH ☐ EDP
II ☐ Professionista/Consulente
LL ☐ Docente/Formatore
MM ☐ Studente
NN ☐ Altro (specificare)

INTERESSI PRINCIPALI
01 ☐ EDP
02 ☐ Personal Computer
03 ☐ Home Computer
04 ☐ Automazione Industriale e Meccanica
05 ☐ Strumentazione elettronica
06 ☐ Telecomunicazioni e telefonia
07 ☐ Elettronica professionale
08 ☐ Elettronica hobbystica
09 ☐ Elettrotecnica e impianti elettrici
10 ☐ Strumenti musicali
11 ☐ Marketing e management
12 ☐ Broadcast/audio e video professionale
13 ☐ Didattica
14 ☐ Altro (specificare)

CHE PERSONAL COMPUTER POSSIEDE
DOS ☐ MS DOS e compatibili
MAC ☐ Macintosh
AMG ☐ Amiga
C64 ☐ Commodore 64
VAR ☐ Altro home computer

NOVITA' ASSOLUTA IN EDICOLA

L. 3.500

LA GRANDE GUIDA A TUTTI I GIOCHI ELETTRONICI E NON

Nuovissima, ricca e tutta a colori. GUIDA VIDEOGIOCHI ti aspetta in edicola con oltre 60 giochi recensiti, i commenti, le curiosità, i trucchi e le novità da tutto il mondo.

E, in più, partecipi al grande concorso riservato ai fedeli lettori di GUIDA VIDEOGIOCHI.

FANTASTICO CONCORSO GUIDA VIDEOGIOCHI

I premi in palio sono favolosi: due esclusive Control Deck NINTENDO e tanti game originali.

GRUPPO EDITORIALE JACKSON

Nintendo

MATTEL

Aut. Min. Rich.